AN ABSTRACT OF THE THESIS OF

I-CHIEN WANG for the degree of Master of Science in Industrial Engineering presented on June 9, 1995.

Title: Conceptual Modeling Architecture and Implementation of Object Oriented Simulation for Automated Guided Vehicle (AGV) Systems.

Redacted for Privacy

Abstract Approved: _____
Terrence G. Beaumariage

Traditional simulation languages and simulators do not fully support the need to design, modify, and extend simulation models of manufacturing systems, especially, material handling systems. Since AGV systems, one type of automated material handling systems, require complicated control logic, flexible job routings, and frequent layout modifications and extensions to correspond to production requirements, the time consumption and efforts to achieve the above tasks in traditional paradigms are significant. However, such difficulties can be overcome by the use of object-oriented simulation.

This research develops an object-oriented modeling architecture for the simulation of AGV (automated guided vehicle) systems by extending Beaumariage's object-oriented modeling environment (1990) which is originally designed for the simulation of job shop type manufacturing systems. For this extension, several classes required to comprise an AGV

system are created into the original environment which include AGV, limited size queue, control point, track segment, machine cell, AGV system control classes, and so on. This architecture provides a flexible environment that enables the modeling of traditional and tandem AGV system layouts. A best-first search approach, one artificial intelligence search algorithm, is employed to direct AGVs to determine the shortest path from all possible travel paths. The computerized modeling system with this conceptual architecture is easy to use, especially compared with traditional simulation tools. In addition, the extended object-oriented architecture used for the simulation of AGV systems is program independent and may be implemented in any object-oriented language.

The prototype system implemented as a portion of this research is performed in Smalltalk/V. Two case examples are presented for verification and validation.

CONCEPTUAL MODELING ARCHITECTURE AND IMPLEMENTATION OF OBJECT-ORIENTED SIMULATION FOR AUTOMATED GUIDED VEHICLE (AGV) SYSTEMS

by

I-CHIEN WANG

A THESIS

Submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed June 9,1995
Commencement June 1996

Master of Science thesis of I-Chien Wang presented on

June 9,1995

APPROVED:

Redacted for Privacy

Major Professor, representing Industrial and Manufacturing Engineering

Redacted for Privacy

Chair of Department of Industrial and Manufacturing Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University Libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

I-Chien Wang, Author

ACKNOWLEDGMENTS

First of all, I want to thank my major professor, Dr. Terrence Beaumariage, for his patience and guidance through this research. Without him I would not have this talent or be where I am today. Then I want to thank Dave Niess who spent a significant amount of time installing and fixing UNIX SLAM system for this research. Without his help, this thesis can not be smoothly completed. I would also thank my Mom and Dad for bringing me into the world and supporting me. And Chun-Ho Kuo for just being there when I need her. She has endured all the hard time with me during my research period.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

LIST OF TABLES

CONCEPTUAL MODELING ARCHITECTURE AND IMPLEMENTATION OF OBJECT-ORIENTED SIMULATION FOR AUTOMATED GUIDED VEHICLE (AGV) SYSTEMS

CHAPTER 1. INTRODUCTION

Designing an automated material handling (MH) system is a complex task. Due to the complexities and variety inherent in MH systems, the use of mathematical or queuing theory based analyses is not sufficient to accurately describe and evaluate the systems. Simulation, especially discrete event simulation, seems to be the only tool which can give a prediction of the MH system performance with a high degree of detail and accuracy (Gong 1990). However, the effort needed to design a simulation model or program for simulating MH systems is enormous. Therefore, simulation software used to design MH simulation models is desired not only to be able to capture details of the system being modeled, but to do so while being relatively easy to use. Unfortunately, conventional simulators and simulation languages can not simultaneously achieve these goals. Thus, a new approach in building simulation models for MH systems must be developed. The application of the object-oriented programming (OOP) paradigm to simulation modeling for MH systems is appealing under this circumstance.

This research addresses the simulation modeling and analysis of automated guided vehicle (AGV) systems. The primary goal of this research is to create a conceptual architecture for the object-oriented simulation (OOS) of the AGV system within an existing OOS paradigm (Beaumariage 1990) after defining the detailed information requirements needed to model an AGV system. The secondary goal is then to implement a prototype system in Smalltalk/V and verify/validate its operations. Furthermore, the benefits of OOP applied to simulation will also be discussed.

1.1 Problem Statement

The nature of the factory has changed from the static environment it once was to the dynamically changing system commonly found today. In recent years, due to the competitive pressure in the market to rapidly increase flexibility and reliability, MH systems of many varieties have been extensively implemented in manufacturing.

Since MH equipment usually costs a large amount of money, in order to achieve the expected performance after installation a detailed analysis is required. Due to the complexity of the problems associated with MH systems, it is intractable to analyze the systems by using mathematical models. Therefore, simulation becomes an excellent tool used in the modeling and analysis of large-scale MH systems.

Typically, simulation software can be classified into two categories, namely, simulators and general-purpose simulation languages.

Simulators, such as ProModel and AutoMod, provide users with a programming free, menu-driven, and graphical interface environment to build simulation models. Simulators are easy to use, understand, and, also, time saving in the development of simulation models. But, in order to achieve these advantages, simulators have to restrict their capabilities, simplify system complexity, and thus lack flexibility. Due to their nature, simulators are only suitable for certain application domains and can not meet all simulation users' needs.

General-purpose simulation languages, such as SLAM and SIMAN, do not have these limitations. The modelers can use the predefined general simulation functions to design their simulation models with a high degree of detail and flexibility. Current simulation languages typically provide a network drawing or flow-chart like environment with numerous standard simulation blocks to allow modelers to approach simulation problems in their own manner. In addition, most simulation languages offer an interface for user-written subroutines in traditional languages which can be used to enhance the ability of the original environment to capture the desired detail in the model. Even though general-purpose simulation languages have such advantages,

there is little doubt that they are difficult to learn and understand especially for users without strong programming skills and simulation background.

However, general-purpose simulation languages and simulators still have such common deficiencies as lack of reusability and extendibility. The users cannot access the internal function of the language or package. Instead, a user must rely on the vendor description of the algorithm, procedures, and data used to implement the concepts. Only the vendor can make modifications to the internal functionality and users have limited opportunity to extend an existing environment.

In order to improve upon current modeling capabilities and paradigms, this research proposes consideration of the object- oriented approach, resulting in greater flexibility, reusability, and extendibility. OOS, which inherits all beneficial characteristics of the OOP paradigm, can result in these improvements. OOS provides a natural framework for development, allowing users to solve a problem in a manner parallel to real-world situations. Furthermore, through the implementation of a modular modeling procedure, OOS can achieve the tradeoff between flexibility and ease of use in simulation software development (You 1993).

1.2 Background of Study

1.2.1 Classifications of AGVs

In recent years, because of the continuous rise of labor costs and enormous market competition, the degree of automation used in manufacturing has increased considerably. Also, since the structure of product markets is trending toward a low-volume and high-variety orientation, batch production and job shop production constitute a large portion of the total manufacturing activities. With improvements such as automatic loading and unloading, the ability to transport a wide variety of material and the capability to interface with other equipment, AGVs have become common in modern manufacturing systems.

"An AGV system is an automated MH system in which driverless, battery-powered vehicles are moved by means of an electronic, chemical, or optical signal from a path installed in or on the floor" (Davis 1986). The movement of an AGV may be managed by a central computer, an onboard computer, or a combination of the two. An AGV system is suitable in applications where different materials must be moved from various load points to various unload points and the routing of material is more individualized (Pritsker 1986).

There are a number of different types of AGVs designed in response to specific production requirements. According to the MH Handbook (Kulwiec 1986), these types can be classified as follows:

- Driverless trains: consist of a towing tractor that pulls trailers on which loads are carried. It is useful in applications where heavy payloads must be moved large distances with intermediate pickup and drop-off points along the route.
- Wire-guided pallet trucks: have a unique capability to move palletized loads along pre-defined routes. Additionally, it can load and unload pallets off the main guidepath to areas where there is no guidepath to follow.
- Unit load carriers: are equipped with some specific devices, such as powered rollers, moving belts, mechanized lift platforms, etc., to automatically load and unload various load configurations between stations. With features such as small floor space, bi-directional travel capability, and variable load capabilities, this type of AGV has become more popular recently (Gong 1990). There are two major categories: light-load and assembly line AGVs. The former is used to deliver small loads through plants. The latter is primarily designed to carry a partially completed subassembly through a sequence of assembly stations.

1.2.2 Characteristics of AGV Systems

- Flexible and automated capabilities. The ease of installation of sealed guidepath wire results in the flexibility of path location of AGV systems. Guidepaths can be rerouted easily and stations can be flexibly added, deleted, or moved. Also, the entire system can be installed in existing facilities or even transferred to a new plant site without major changes in current layout. In addition, AGVs typically are equipped with several kinds of automated devices such as power rollers, moving belts, etc.
- AGVs can be integrated with other types of automatic equipment to enhance the capability of the system. For example, through placement of a robotic manipulator on an AGV, the robot can provide more mobility to perform complex handling tasks (Groover 1987).
- The interactions of the AGV system with other manufacturing system components are intimate and complicated. Especially when the AGV system is an integrated part of a large system, the AGV system performance is directly related to the performance of the total system. For example, "few vehicles may not be able to remove material from stations in a timely manner. On the other hand, adding too many vehicles may cause high levels of congestion which may prevent the vehicles from effectively removing material" (Pritsker 1986).

- AGV systems employ many control policies to direct the movement of an AGV. For example, vehicle selection rules, vehicle dispatching rules, and staging area selection rules.

1.2.3 Characteristics of AGV System Simulation

In the simulation of simple manufacturing systems without the need for installing complex MH facilities or where MH resources are ample, MH functions could be represented simply or ignored completely. However, as the concern for MH functions grows, such simplifications are not usually the case in modern manufacturing situations because there is a limit on the amount of MH resources and the waiting time is very large. The simulation of AGV systems includes the following characteristics:

- The process routing of parts and system layout tend to be frequently changed. Thus, in order to be responsive to the flexible, dynamic, and frequently-changed nature of modern manufacturing environments, simulation of AGV systems normally requires not only frequent extensions, but also iterative modifications.
- A huge number of parameters used to direct the system run need to be specified. As described previously, each type of AGV has specific functions and particular features.

Thus, in order to closely describe AGV behaviors, a simulation system needs the capability to allow users to specify the parameters for their particular designs.

- Several kinds of simulation events need to be considered. For example, AGV breakdowns, AGV maintenance, and battery charging activities, etc. To closely resemble realistic conditions, an AGV simulation should have the capability to model these events.

1.2.4 Simulation of AGV Systems in General-Purpose Simulation Languages

A general-purpose simulation language is defined to be a versatile, general purpose class of simulation software that can be used in a variety of different modeling applications by means of some specific concepts, statements and modules for representing the state of a system (McHaney 1991, Pritsker 1986). Generally speaking, simulation models written in general-purpose simulation languages begin with a network-like structure or flow diagram that graphically portrays the flow of entities through the system. This network-like structure consists of specialized symbols and branches. Symbols are typically used to represent the simulation resources or common simulation functions. Symbols are connected by branches that are used to define the time delay or routing of the entities through the system. Routing may be deterministic or probabilistic, and

time delays on activities may represent processing, traveling, or waiting times. In this section, building simulation models of AGV systems in general-purpose simulation languages is described. The simulation languages that are discussed includes SLAM and SIMAN.

· **SLAM:**

SLAM, the Simulation Language for Alternative Modeling, a product of Pritsker Corporation, has been used since 1979, and was enhanced by the introduction of SLAMSYSTEM in 1988. There are 23 basic specialized symbols called nodes available in SLAM which can be selected from a graphical palette which is part of the network builder window. The entire modeling process of SLAM consists of choosing these nodes which can represent system processes, combining these nodes with branches and parameterizing them with proposed data (O'Reilly 1993). The general features of SLAM are thoroughly covered by Pritsker (1986) and, therefore, are not presented here.

The modeling of AGV systems in SLAM is done by a Material Handling Extension (MHE) which provides several MH resources (cranes, AGVs, etc.). For modeling an AGV system, the MHE provides three types of new resources: VCPOINT, VSGMENT, and VFLEET which are used to represent the physical layout (control points and segments) and describe the behaviors of AGVs, respectively. Control points containing all intersection and load/unload points are described by the

VCPOINT resource in which reference numbers, rules for contention and routing decisions are specified. The VSGMENT resource is used to define track segments between control points. It includes the information of flow direction, length of the segments, and the maximum capacity which allows AGVs to travel on the segment concurrently. The VFLEET resource is used to define a fleet of AGV units. The definition includes a vehicle fleet's physical attributes, next job selection rule, rule for idle disposition logic, and the initial location of the individual vehicle units.

Moreover, three new network nodes associated with the movement of vehicles are included. The VWAIT node is used to model a request for transport by a vehicle from a control point. In the VWAIT node, the file storing waiting entities, the control point's reference number, the vehicle selection rule, and the entity release rule are specified. A VFREE node is used to free a vehicle that has been allocated to an entity. In the VFREE node, the rules for job request and idle vehicle disposition are specified. The actual movement behavior of a vehicle is accomplished by the VMOVE node where the destination control point is specified. A detailed simulation model of an AGV system written in SLAM can be found in Pritsker (1986).

· **SIMAN:**

SIMAN V is a general purpose SIMulation ANalysis program that was introduced in 1982. Models in SIMAN are broken into two parts: the system model and the experimental frame. The system model defines the static and dynamic characteristics of the system such as machines, queues, transporters, etc. The experimental frame defines the conditions under which the model is to be run, i.e., the parameters, priority rules, data, etc.

SIMAN has added the capability of modeling material handling features that can be applied to model two categories of automated MH equipment: transporters and conveyors. Transporter devices may be of two types: free-path or guided. Free-path transporters include forklifts, carts, platform trucks, and so on. Guided transporters include AGVs, industrial cranes, AS/RS, etc.

When modeling an AGV system, the network flow in the system model part contains the blocks with such functions as allocating AGVs to an entity, moving AGVs from one station to another, changing the status of AGV from busy to idle, making an AGV unavailable/available, etc. The elements in the experimental frame regarding the AGV system define the characteristics of the AGV system resources, intersection points, links, networks, etc. A detailed simulation model of an AGV system written in SIMAN can be found in Banks (1990).

1.2.5 Material Handling System Simulation in Simulators

A recent development in simulation application is simulators. A simulator is defined as a user friendly software that will develop a simulation model for a particular application (McHancy 1991). The principal idea simulators apply is that a simulation model's structure is built by using the same familiar process one would naturally use to describe a real system. More specifically, when modeling a flexible manufacturing system, modelers may begin by defining the flow of parts through the system, and then specifying what operations are performed at each location. A location may be a machine, work station, queue, or a position on a conveyor or transporter path. Once the routing for each part type has been defined, modelers schedule the arrival of parts to the system and then specify resource capacities and any special operating characteristics of the resources being used. In simulators, each of these tasks is completed typically using easy-to-use building blocks that are automatically linked by information supplied in a previously completed building block. The simulators discussed in this section include ProModel and AutoMod that are specifically applied to the modeling of manufacturing systems.

· **ProModel:**

ProModel (PROduction MODELer), a product of Production Modeling Corporation of Utah, is a typical modularized simulation and animation tool that provides a pop-up editor interface allowing users to quickly simulate manufacturing systems.

ProModel's building blocks are called modules. There are two major modules called model definition module and model layout module that are used to define the logical and physical components of the system being modeled, respectively. The model definition module contains several submodules to define the logical part of the simulation system such as part routing, part scheduling, part information, resource capacities, downtimes, material handling resource definitions, and simulation parameters. The model layout module is used to define the information needed to properly display simulation graphics, represent static and dynamic model entities, and construct the layouts of material handling devices. The submodules contained in this module are graphic option, entity symbol, tranporter path, conveyor layout, and icon editor modules. Essentially, ProModel is designed to allow simulation models to be built in modules and then merged to form one model.

ProModel's built-in material handling capabilities enable users to easily and realistically model complex operating characteristics and control logic of automated MH systems such as robots, AGV systems, conveyors, and AS/RS.

The modeling of AGV systems is included in the transporter submodule under the model definition module mentioned previously. The transporter module actually consists of four submodules which are used to define the operating characteristics of each transporter system. "The transporter specifications submodule defines the operational characteristics of the transporter system such as initial location, speed, and job search priority, etc. The transporter path logic submodule defines the point connections and path block which make up the path logic for a transporter. The location interface submodule identifies the linkage between points and location interface points for the transporter system being defined. The transporter search priority submodule provides a mechanism for assigning priorities to the transporter's search for work, a place to park when idle, or the preferred route from one location to another" (Production Modeling Corporation 1992). A detailed simulation model of an AGV system written in ProModel can be found in the ProModel User Manual (Production Modeling Corporation 1992)

· **AutoMod:**

AutoMod, a product of AutoSimulations Inc., is an industrial oriented simulation system that offers numerous advanced capabilities such as easy-to-use graphical pop-up menus, spreadsheet interface, and CAD-like drawing tool that can produce accurately scaled, 3-D animated manufacturing

models to reflect the logical and physical aspects of systems. In AutoMod, the building blocks discussed previously are called data tables by which users can describe the components of a manufacturing system such as resources (machines, people, and tools), processing steps (part routing), and production quantities (orders), etc. AutoMod contains built-in material handling templates allowing users to simulate a wide variety of material handling devices such as conveyors, AGV systems, AS/RS, towlines, bridge cranes and robotic mechanisms.

When developing simulation models of AGV systems with AutoMod, there are some important issues concerned with movement control, collision avoidance, and task assignment. AutoMod provides powerful yet easy-to-use features for describing guidepath, control points, work search algorithms, and vehicle park rules which direct the movement control of AGVs. AutoMod automatically models vehicle acceleration and deceleration, and calculates the shortest path for vehicle routing such that the user is not required to define the paths for each of the possible vehicle trips. Also, AutoMod provides very sophisticated blocking and collision avoidance features. AutoMod allows nine different work search schemes to deal with the work search algorithm. These nine may be combined for more complex searches. A simulation model of an automated MH system written in AutoMod can be found in Banks (1990).

1.2.6 Object-Oriented Simulation

Object-oriented simulation (OOS) is the application of object-oriented programming (OOP) to simulation which is a different philosophy of design from traditional procedural programming. The main concept of OOP is that it conforms to the notion that the world is composed of different "objects" rather than programs and data files. Objects are intended to represent the components of the system. Like the components in the real-world system, each object has its own attributes, behavior, and interaction with other objects. Thus, through identification of objects and appropriate communication between different objects, an object-oriented system can be built and then function.

The basic structure of object-oriented world consists of objects, variables, methods, messages, classes, etc. As I have mentioned previously, all items in the system are treated as objects. An object is an abstract data type in which the private variables and methods associated with the object's characteristics and behaviors are included. In object-oriented terms, we say that the variables and methods are encapsulated within the object. Variables are an object's private data used to represent its attributes. Methods are the algorithms that determine an object's behavior and performance. Methods are somewhat like function definitions in other traditional languages. If an object desires to communicate with other objects to obtain

some information about their state or respond to requests from other objects, it must send and receive appropriate messages. In object-oriented languages, objects communicate to other objects just like people communicate in our real world. Classes are software modules that define a set of objects with similar characteristics and behavior. Actually, thus an object can be either a class or an instance of a class. Normally, classes are specified in a hierarchical tree structure defining their relation to other classes in the system. In this manner, a lower hierarchy object can inherit the generic properties and behavior from its parent classes. OOP embodies five key features which result in making a software system more understandable, modifiable, reusable and flexible.

• Encapsulation:

An object's variables are enclosed within a controlled and tight boundary along with the methods which are able to use the data. This enclosed boundary defines the object and protects the data from unauthorized external access. In other words, variables stored within an object are directly accessible only by the methods that have been defined as part of the class to which the object belongs. Two major benefits of encapsulation are understandability and modifiability. First, objects are clearly and completely defined because "their variables and methods are implemented in a coherent manner rather than the loose combination of

multiple procedural routines" (Mize 1992). Secondly, the system is more modifiable because a change to one part of the system does not force unanticipated changes to other parts of the system.

- Message passing:

This is the way objects communicate with one another. Since all variables and methods of an object are tightly encapsulated, in order for one object to request information from another object, the first object must send a message to the second object to trigger the execution of a method regarding the desired information.

- Inheritance:

OOP classes are defined and arranged in a hierarchical tree structure. Each class can inherit the variables and methods defined in all its super classes and can also acquire it's own characteristics. The benefits are reusability and reduction of code size because the instances of existing classes can be reused to build other new classes or reused in other applications.

- Polymorphism:

Polymorphism is the ability for different objects to respond to the same message in their own way. For example, trucks and cranes may both have the ability to pick up and deliver parts. We can apply the same methods "pick-up" and "deliver" to both truck and crane classes to perform the same function.

• Late binding:

Late binding or dynamic binding means that the attributes of expressions or types of variables are determined at run time rather than at compile time. This feature provides for variable types to change during execution. Thus, it allows users to flexibly change the variable types if necessary.

The application of the object-oriented paradigm to simulation is a relatively new approach. In essence, it changes the way users think about modeling the system of interest. For example, when describing a manufacturing system, people intuitively think of it in terms of workers, machines, jobs, routings, conveyors, and so on. However, when we begin to create a simulation model of this system, supposing in a general-purpose simulation language, we may find that the situation which we describe in terms of objects now must be translated into some other different terminologies such as resources, queues, activities, etc. Thus, it is more convenient if we can design the simulation model in a manner similar to the way we think of the actual system. Due to the features I have discussed before, OOS easily achieves this goal. OOS users can create new objects in response to users demands and build them into a consistent platform. In addition, OOS provides the simulation user with a convenient environment to build a prototype system in an intuitive fashion.

1.3 Literature Review

1.3.1 Recent Studies on AGV Systems

There are three major components comprising an AGV system. Several eminent studies associated with each component will be briefly discussed in this section.

• Vehicles:

One of the important issues about vehicles is to determine the minimum number of AGVs required in a given system. Maxwell and Muckstadt (1982) developed a simple time-independent model to find the minimum number of AGVs. A dynamic evaluation of vehicle requirements is then conducted to determine whether or not the suggested number of vehicles can carry out the movement requirements. Later, Egbelu (1987) presented four simple analytical models to estimate the number of vehicles required. Tanchoco *et al.*(1987) determined the total number of vehicles in an AGV-based material transport system.

• GuidePath:

The guidepath system can be classified into two distinct areas: path flow and path layout. As to path flow, Egbelu and Tanchoco (1986) analyzed the possibility of bi-directional flow in guidepaths. Based on a simulation

analysis, they concluded that bi-directional flow reduces the travel time of AGVs, increases system throughput, and improves machine utilization. Fujii and Sandoh (1987) used graph theory, combinatorial analysis, and linear programming techniques to develop a routing algorithm which minimizes the total traveling time of all AGVs. Kim and Tanchoco (1991) proposed a conflict-free, short-time algorithm for bi-directional AGV routing. Dhouib and Ait Kadi (1994) developed an expert system prototype for collision avoidance and routing uni-directional AGVs in bi-directional networks.

As to path layout, Maxwell and Muckstadt (1982) developed an AGV system layout served by multiple vehicles that is called traditional multiple vehicles layout today. The traditional AGV system is shown in Figure 1. In this AGV system, each vehicle is allowed to visit any pick-up/deposit points. Thus, the routing is flexible, but the drawbacks are many control points are implemented and collisions and blocking problems may frequently occur. Before about 1990, almost all research work on AGV system layout or path flow were based on the traditional layout. For example, Gaskins and Tanchoco (1987) formulated the guidepath layout problem as a zero-one integer programming model under the assumption that the facility layout and pick-up/deposit stations exist already. Usher *et al.*(1988) proposed a procedure to locate the pick-up/deposit stations under a similar assumption.

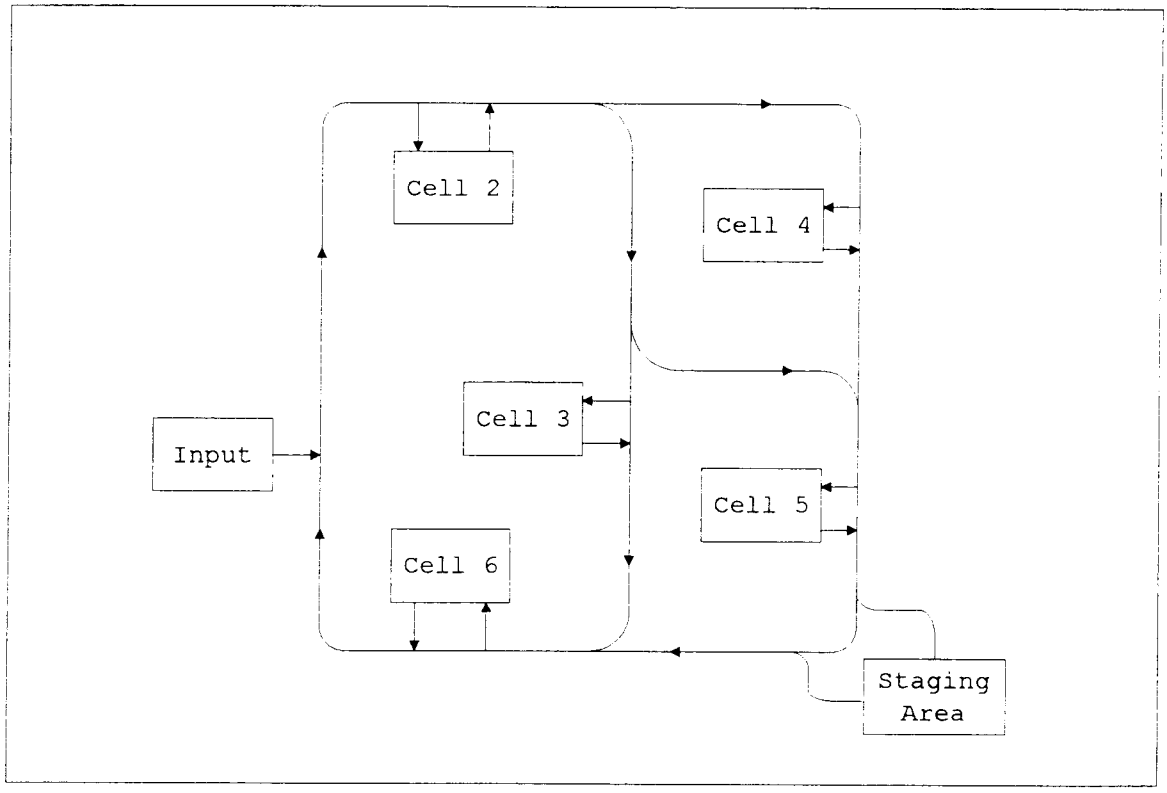


Figure 1: Traditional AGV system layout

• Control logic:

The third major component of an AGV system is control logic. Egbelu and Tanchoco (1984) examined the effects of several heuristic rules for dispatching AGVs in a job shop type environment. The study considered the case where the material flow volume is large. Barthodi and Platzman (1989) developed the First-Encountered-First-Served(FEFS) heuristic from the point of view of a single AGV to design a single loop AGV system with decentralized control logic. Simulation results of this study were consistent with the

result that FEFS works quite well for an AGV traveling in a single loop.

1.3.2 Tandem AGV Systems

Barthodi and Platzman (1989) triggered the modern design of AGV system layout by applying an idea of decentralized control over AGVs to a single AGV traveling loop. Compared to the centralized control used in the traditional layout, single loop AGV systems with decentralized control not only result in less part waiting time, but also each AGV can be programmed identically and independently of the others. In the meantime, Bozer and Srinivasan (1989) introduced a conceptually simple and intuitive approach to design the AGV system layout called tandem configuration where the system is decomposed into non-overlapping, single-vehicle closed loops with specified stations provided as an interface between adjacent loops. An example of a tandem AGV system is shown in Figure 2 where stations correspond to those in figure 1 except the staging area. Note that each AGV is assigned to only one loop and each loop has only one AGV. Tandem configuration not only eliminates traffic congestion problems, but also needs significantly less complicated vehicle dispatching and traffic management due to the fact that only a single vehicle is dispatched over a smaller number of stations. In

addition, the tandem configuration also supports the effective use of bi-directional vehicles and offers more flexibility. Choi *et al.* (1994) used simulation models to evaluate the traditional layout and tandem layout by testing various system design and operation parameters. They concluded that the tandem layout could produce more job completions and the traditional layout, on the other hand, may generate lower average throughput time, higher average AGV utilization, lower AGV idle time, and lower average waiting times at the staging area.

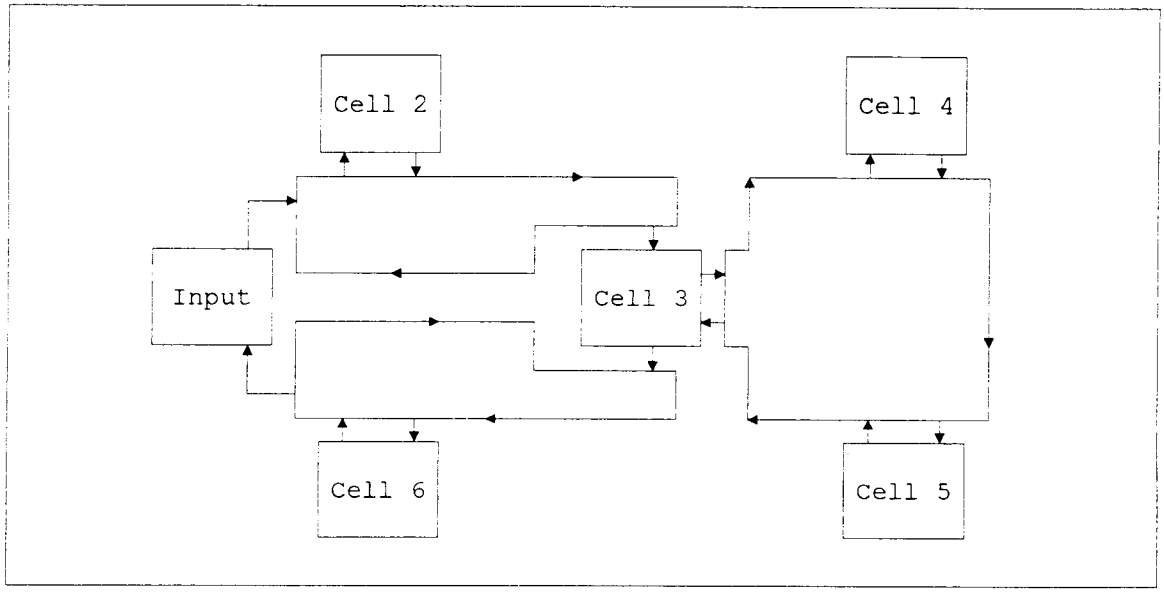


Figure 2: Tandem AGV system layout

1.3.3 Research on AGV System Simulation

In recent years, many researchers have used simulation to analyze and evaluate the performance of AGV systems. In addition to those developed for analysis of previously mentioned issues, Ozden (1987) did a simulation study written in LISP to investigate the effect of several key factors related to multiple-load-carrying AGVs in a flexible manufacturing system. Gaskin and Tanchoco(1989) developed a flexible C-based discrete event simulator so that many system configurations and bi-directional flow can be modeled. Mahadevan and Narendran (1990) constructed an AGV-based MH system model for an FMS using GPSS and concluded that the single vehicle loop configuration and the sequential dispatching rule are better than the other rules in the system considered for study. Lee *et al.*(1990) developed an AGV simulator in SIMAN to study an assembly system with the traditional AGV layout.

But, as we have realized, using conventional tools to simulate MH systems has some specified disadvantages that I have discussed before. In order to alleviate those drawbacks, simulationists have developed several new design approaches. For example, Gong and McGinnis(1990) used a simulation code generator (SCG) written in Quick Basic to generate a SIMAN simulation program for AGV systems. The basic concepts of SCG are that the data required to build a simulation model, provided by the user, is collected by a

data-driven interface (generator) and stored into a database in file form. The code generator will automatically retrieve and transfer this data into a target simulation language by a number of processes. Finally, the simulation is run and statistics are obtained. Another recently developed design approach is OOS. There have been several significant OOS researches for modeling manufacturing systems but not specifically focus on modeling AGV systems. For example, Roberts *et al.*(1992) designed an object-oriented modeling environment for manufacturing systems with C++ and Tanchoco (1993) proposed a modular framework for the design of a material flow system, etc.

1.4 Research Goals and Objectives

The principal goal of this research is to develop an object-oriented modeling environment architecture for the generation of simulation models of AGV systems by extending Beaumariage's paradigm (1990). In order to achieve the above goal, the objectives of this research are to:

1. Define the detailed information requirements needed for a modeling system for AGV systems.
2. Create a conceptual architecture for the OOS of AGV systems within Beaumariage's paradigm (1990). This conceptual architecture is program-independent and can be

implemented in any object-oriented language.

3. Implement a prototype system in Smalltalk/V and verify/validate its operation with a simple system written in both this prototype system and SLAM.
4. Determine a case study and then measure and analyze the contained AGV system using the Smalltalk prototype system.

The prototype simulation environment for AGV systems implemented in this research has the capabilities of uni-directional guidepath, multiple vehicle staging areas, resource reservation, vehicle breakdowns, different types of vehicle velocities, and vehicle battery charging capability that allows the users to design the proposed AGV system with either traditional or tandem layout. In addition, vehicle selection and dispatching rules have several options to choose. The simulation outputs of this system include the general performance measures such as system throughput, machine cell statistics (utilization, queue information, etc.), AGV utilization, track segment utilization, and control point utilization.

CHAPTER 2. DESIGN OF AGV SYSTEMS

When designing an AGV system, several components cooperating to accomplish the entire performance must be considered. Generally, they fall into three major categories.

2.1 Physical Configuration

Physically, an AGV system is comprised of guidepaths, work cells and vehicles. I have already discussed vehicles previously. Guidepaths are closely related to the production system. Typically, the production system constructed with an AGV system is assumed to be a job shop type which has several work cells performing different operations. Each work cell consists of a number of identical machines to process various types of parts which have different sequences of processing routes. The AGV guidepath is constructed to link work cells such that parts can be transported through the system by AGVs. The detailed descriptions of physical configuration of AGV system are:

- Machine cells, and control points:

  A machine cell is represented as a block in which parts are received and released through the guidepath. AGVs only transport parts in and out of machine cells.

In other words, each machine cell may have its own transport device to handle its internal part movement, and this internal movement is independent of other machine cells. Furthermore, each machine cell may have many separate or combined load/unload stations at which parts can be loaded or unloaded from AGVs. Load/unload stations and the intersections of track segments are usually specified as control points.

- Track segments:

The guidepath is constructed by the segments that connect control points. Each track segment is non-overlapping and capable of providing either uni-directional or bi-directional operation. For control purposes, track segments are allowed to contain a limited number of AGVs. Normally, this limited number is one.

- Vehicle staging areas:

Most AGV systems employ a staging scheme, such that all idle vehicles are sent to a common area, where they await the next tasks to perform. If the system doesn't have such a staging area, an idle vehicle without any tasks to perform may stop at any location in the system, blocking the guidepath, and disrupting system performance. In addition, the staging areas can

be designed as battery charging areas to conduct full-cycle battery charging operations. Any AGV that is nearly depleted of its charge will be sent to these areas. Figure 3 demonstrates the physical configuration of a simple AGV system.

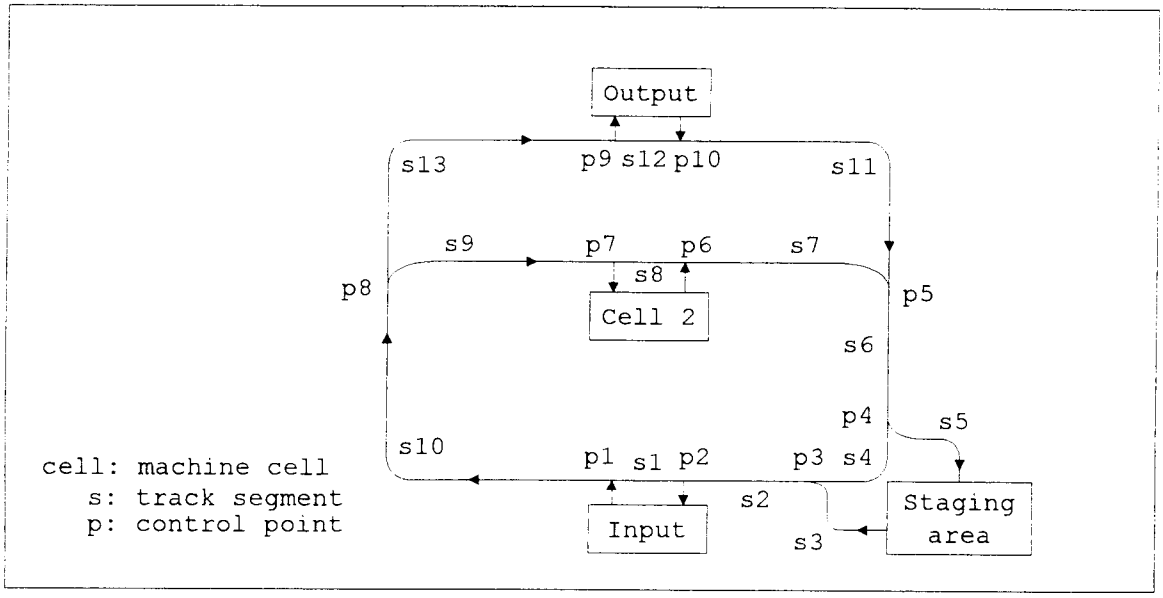


Figure 3: Physical configuration of an AGV system

2.2 Control Capabilities

- Traffic flow control:

The movement of AGVs can be designed in two ways, namely, uni-directional and bi-directional. When an AGV is moved from one station to another by either one of the two types, the vehicle must know what path it should follow. Typically, it will attempt to take the

shortest path to its destination. Due to this, it has resulted in a need to maintain the physical component location, from which the route between stations may be determined.

- Prohibition of collision and traffic control:

If a single AGV operates in a closed loop, the traffic control problem is simple. However, once multiple vehicles exist in the system, the problem of vehicle congestion must be solved by means of control zones(track segments and control points). To avoid collision, each control zone is normally permitted to be occupied by only one vehicle at a time. When an AGV starts its movement to the next track segment or control point, it must check if there is another AGV occupying it. If another AGV is utilizing the control zone in front of its current location, it must stop and wait until the next zone is clear before it can continue its journey.

- Vehicle selection rules:

When a job arrives to the system or completes processing at a station, an AGV will be assigned to pick up this job if there are AGVs available. If there are none, the job enters the queue associated with the machine station and awaits assignment until one AGV is

available. If only one vehicle is available, it is definitely allocated to that job. However, if more than one AGV is available at the same time, there is a need to use a "vehicle selection" rule to determine which one should be assigned to that job.

The most commonly applied is "Nearest Vehicle" which assigns the AGV with the shortest travel distance to the job's location. In essence, this rule attempts to move jobs through the system as quickly as possible (Davis 1986). Another commonly used rule is "Lowest Utilization Vehicle" which attempts to equalize the utilization of each AGV. Each of these assumes that the AGVs in the system are identical. However, if, based on economical considerations, we implement nonidentical AGVs in the system, the vehicle selection rule will be somewhat more complicated. For example, when a job is waiting for transport, the system needs to make sure not only that a vehicle is available, but that the available vehicle has sufficient capacity to deliver the job.

- Vehicle dispatching rules:

After an AGV unloads a part and then becomes idle, it could be immediately assigned the next task to perform, or move to an appropriate staging area to wait. The vehicle dispatching rule is used to decide, if there are many parts waiting for transport, to which

part the vehicle should be assigned. Several rules may be used to provide decisions. Two commonly used ones are "Earliest Queue Arrival Time" and "Nearest Distance". "Earliest Queue Arrival Time" is intended to shorten the time jobs spend waiting in the output-queue, while "Nearest Distance" tries to minimize the idle duration of AGVs. Again, these assume that we have identical vehicles. Suppose that we have nonidentical AGVs in the system, any AGVs ready for the next task need to make sure whether they have sufficient capacities to deliver the jobs.

2.3 Other Advanced Control Capacities

The issues described in previous sections are the primary concepts to direct AGV movements. To resemble the real-world systems more closely, a number of advanced control capacities may be employed.

- Resource reservation capabilities:

  When a part completes processing at a machine cell and hits the end of the output-queue, before it goes to the next machine cell in accordance with its process route, it needs to reserve a space either from the servers or input-queue. Once the reservation is made, it can ensure that when this part arrives to the unload

station, the machine cell has available space to provide processing. Otherwise, the AGV carrying this part will block the guidepath and disrupt system performance. Similarly, once a part finds an appropriate AGV that is able to transport it, the part must reserve it immediately to avoid the chance that this AGV will be requested by another part later. In addition, when an AGV needs to go to staging areas to either perform battery charging or wait for next tasks, it must make a reservation so that other AGVs are not permitted to utilize the station.

Resource reservations are really important for the efficient performance of AGV systems because it not only keeps the whole system utilized but also avoids traffic congestion.

- Vehicle breakdowns:

In a real-world AGV system, for some reasons, such as human errors, track failure, etc., AGVs can break down anywhere in the system. So, a maintenance duration is required to resume the failed AGV. An advanced simulation system should have the capability to schedule both breakdown and maintenance events specified by a specific distribution.

- Vehicle velocity:

AGVs are commonly assumed to have a constant

velocity for all movements within the system. However, in practice, this is not always the case. AGVs travel at a faster rate when they are unloaded than when they are carrying parts. Also, an AGV may decrease its velocity as the battery wears down, or as it approaches a station or turn. In order to respond to these practical situations, an advanced simulation system should be able to provide such types of velocities when AGVs travel along the guidepath.

- Battery charging:

Since movements, loading and unloading operations cause battery consumption, AGVs need to be charged. In practice, battery charging is usually accomplished by one of two techniques: opportunity charging or full-cycle charging. The opportunity charging technique is that the batteries are charged while the AGV is performing or waiting to perform a task. Control points at which the AGV stops for any duration may be the possible charging stations. The full-cycle charging technique requires the AGV to pull itself off the main guidepath and go to a charging area once the battery is nearly depleted of its charge.

CHAPTER 3. STRUCTURE, CONCEPTS, AND IMPLEMENTATION FOR THE OOS OF AGV SYSTEMS

3.1 Original Object-Oriented Modeling Environment

This research extends Beaumariage's (1990) object-oriented modeling (OOM) environment to include the representation of AGV systems. The original environment was developed for the generation of simulation models of job shop type manufacturing systems. It provides a basic and extendible structure for an OOS environment. The class hierarchy of this modeling environment is shown in Figure 4.

All objects (classes) in this modeling environment are arranged in a hierarchical tree structure under the root class SimObject which is a subclass of Smalltalk's highest class Object. That is, class SimObject involves all subclasses to define the simulation world. Each new class that may help users model the system of interest must be defined as a subclass of SimObject. This structure fully reveals the major benefit of OOP: inheritance. Classes higher in the hierarchy represent more general characteristics, while classes lower in the hierarchy represent more specific characteristics. Instead of creating extra code, the users can reuse all generic characteristics from higher hierarchy objects. Thus, the modeling structure

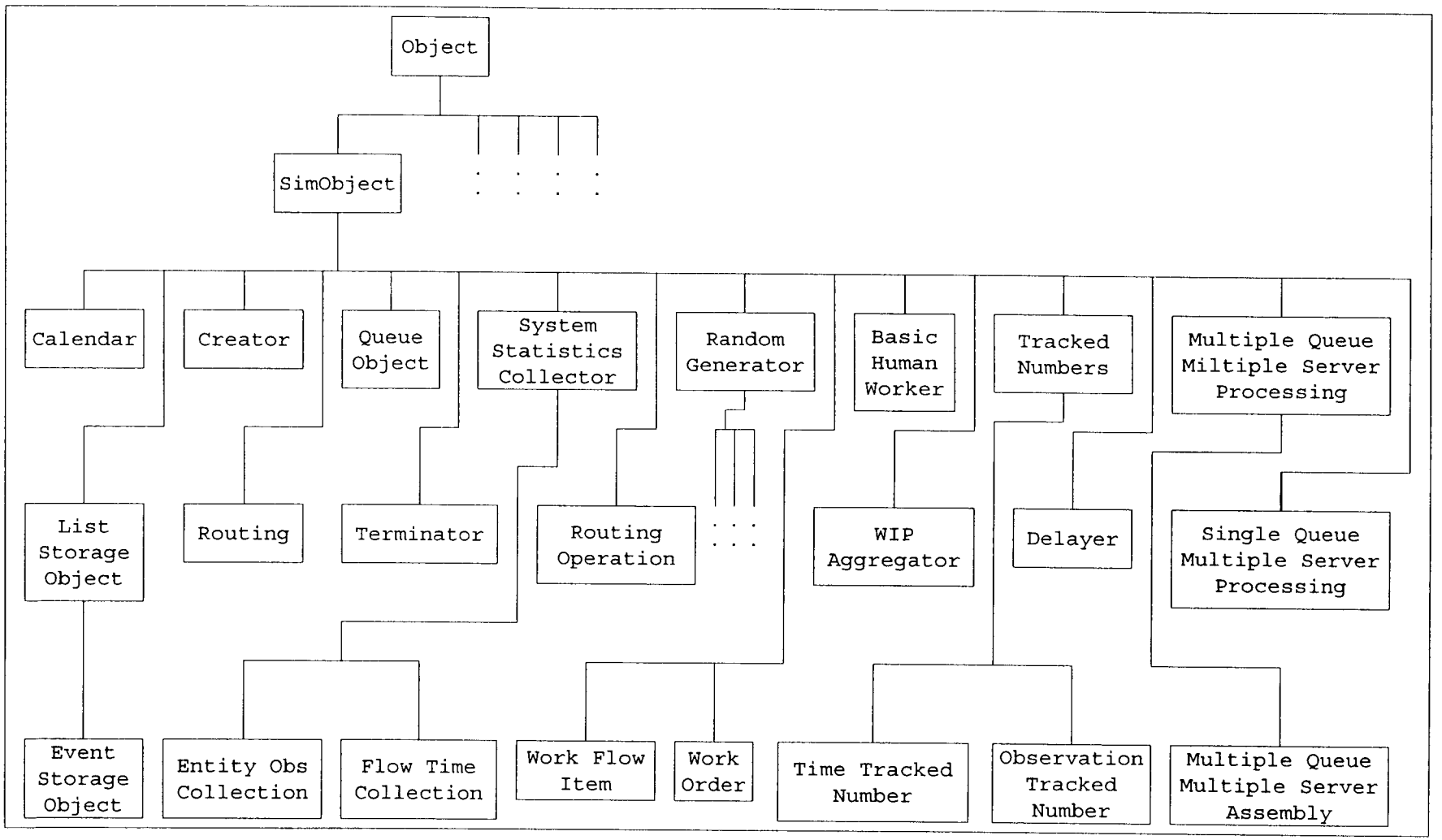


Figure 4: Overall structure of Beaumariage's OOS environment [from Beaumariage(1990)]

with this design manner can considerably reduce code size and coding effort. In addition, it can enhance the readability and modularity of the overall modeling environment.

The construction of Beaumariage's OOM environment can be classified into two different objects: simulation processing objects and simulation element objects. According to Beaumariage's explanation, "Simulation processing objects are abstract objects providing the software functions which allow the background simulation processing tasks, such as time advance, event triggering, entity creation, list processing, etc., to be performed. Simulation element objects, which provide the reusable simulation model building blocks, such as queue objects, machine objects, etc., are implemented in such a way that their actions model the activities of actual elements making up the system of interest" (Beaumariage 1990).

This research will reuse most of these existing objects and execute limited modifications to meet the requirements of modeling AGV systems. Simulation processing objects will be considerably reused since they provide the necessary mechanisms of simulation. The new extended objects created to support AGV system operations will belong to simulation element objects. Such adjustments do not affect the original capabilities, but actually add additional features.

3.2 Modifications of the Original Modeling Environment and Research Assumptions

Beaumariage's modeling environment involves all necessary objects providing both background simulation operations and representation of activities of actual elements making up a general job shop type manufacturing system. However, this environment does not include material handling devices. Also, transportation and queue resources are simply assumed to be ample. Thus, in order to make this research manageable, numerous modifications and additional assumptions are necessary.

3.2.1 Modifications of the Original Modeling Environment

- Machine cell structure:

The original environment allows machine cells to only have input-queues. In the AGV systems of this research, each machine cell is assumed to have one input-queue and one output-queue that are used to temporarily store parts to process or transport to another machine cell. Additionally, each queue is associated with a loading or unloading station or both input and output queues of a machine cell can share only one combined loading/unloading station. The structure of machine cells in the AGV systems is shown

in Figure 5.

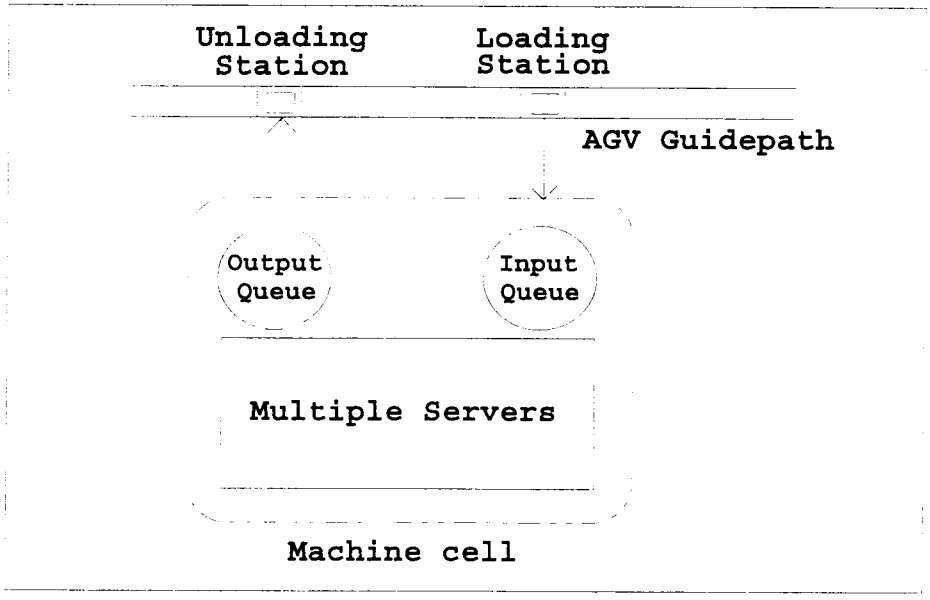


Figure 5: Structure of machine cells in the AGV systems

- Limited queue size:

In this research, the capacity of each queue of the machine cells is assumed to be limited. The original modeling environment allows only unlimited queue size. Although availability of unlimited size queues is reasonable for job shop type manufacturing, provisions for limited queues is closer to the situations of real AGV systems.

- Availability checking at machine cells:

Since the capacities of machine cells (input-queues and servers) in the AGV systems are limited, the simulation system has to check for the availability of the next machine cell before transferring a part from the current machine to that machine. If we do not consider this situation, a part may have no place to go after it leaves the current station. However, in the original environment, we don't need to consider this situation because the input-queue size is unlimited. Once a part arrives to a machine cell that is busy, we can send this part into the input-queue.

- Part reservation at machine cells:

As I have discussed previously in section 2.3, resource reservation is a very important capability in the simulation of AGV systems. As soon as a part makes sure that there is available space in the next machine cell (either from servers or input-queue), it must make a reservation immediately to ensure that it will absolutely have a space when arriving to that machine. However, in the original environment, similar to the availability checking situation, if a part has no space in the servers after arriving at machine cell, it can go to the input-queue.

- Parts balking from input machine cells:

The machine cells in the AGV systems can be classified into two different types: input stations and single input/output queue, multiple server processing objects. The differences between these two types of machine cells will be discussed in section 3.3.1.
The extended AGV system environment allows the balking of parts from the system through input stations. When a part created by creators is sent to the input station but the input-queue associated with this input station is full, then this part will be balked from the system. Similarly, in the original environment, Beaumariage did not design this mechanism due to the unlimited queue resources.

3.2.2 Research Assumptions

1. Each machine cell contains a number of identical machines (servers) performing the same manufacturing processes.
2. AGVs only transport parts in and out of the machine cells. Each machine object may have its own device to handle internal part movement. All internal part transport time will not be counted.
3. The part transport time between guidepath and queues of machine cells will also not be counted (refer to

Figure 5).

4. AGVs only deal with transportation of parts without integrating with other manufacturing equipment.

5. The process route of each part is deterministic. Multiple route cases are not considered.

6. At most only one AGV can occupy a control point or track segment at a time.

7. AGVs do not pass one another. Once an AGV's traveling route is blocked by another AGV, the rear AGV will stop at the current control point if the front AGV is in a track segment, or stop in the end of segment if the front AGV is in a control point, until the front AGV leaves.

8. AGVs will attempt to take the shortest path to travel through the system.

9. The length of each control point is assumed to be zero.

10. Bi-directional travel is not considered.

11. AGV breakdowns are statistically scheduled during the simulation.

12. AGVs may be in exactly one of the following states:

- stop
- moving empty
- moving loaded
- charging

13. Only the first part in an output-queue of a machine cell can start moving out of the current

machine cell by checking the next machine's availability and then finding an idle AGV for transport.

14. The determination of which part waiting to transport is based on the "Earliest Arrival Time to Output-queue" rule.

15. When triggering an AGV waiting at the end of input-segments of a control point to resume it traveling, a FIFO rule is used.

16. An idle AGV without any tasks to perform or an AGV needing to charge its battery always selects the nearest available staging area.

17. The battery charging technology used in this research is restricted to full-cycle charging.

18. If the capacity of an AGV's battery is lower than 20% of its maximum capacity, it requires battery charging.

19. Based on the acceleration and deceleration of an AGV, if the length of track segment is not long enough to allow the AGV to travel at its maximum speed, a constant speed of 50% of its maximum speed is assumed.

20. Curve speed of an AGV is not considered.

21. The design of AGV systems in this research is specifically focused on the application of unit load carriers.

3.3 Conceptual Architecture for the OOS of AGV Systems

3.3.1 Extended Structure of AGV System simulation

For the extension of AGV system capabilities, several new classes are added into the original modeling environment. The extended structure of these classes is shown in Figure 6. These new classes belong to the simulation element objects. I add a new class called MaterialHandler under the root class SimObject. The MaterialHandler class is an abstract class which provides the generic characteristics of material handling devices.

Summary of the MaterialHandler Class:

Function:

To provide general characteristics of material handlers.

Data Storage:

- the name of the material handler

A principal class among these new added classes is the class called AGV. It is created under the MaterialHandler class since AGV is one type of material handling device. To resemble the AGV activities of the real system, the AGV class provides several capabilities/methods which include the ability to:

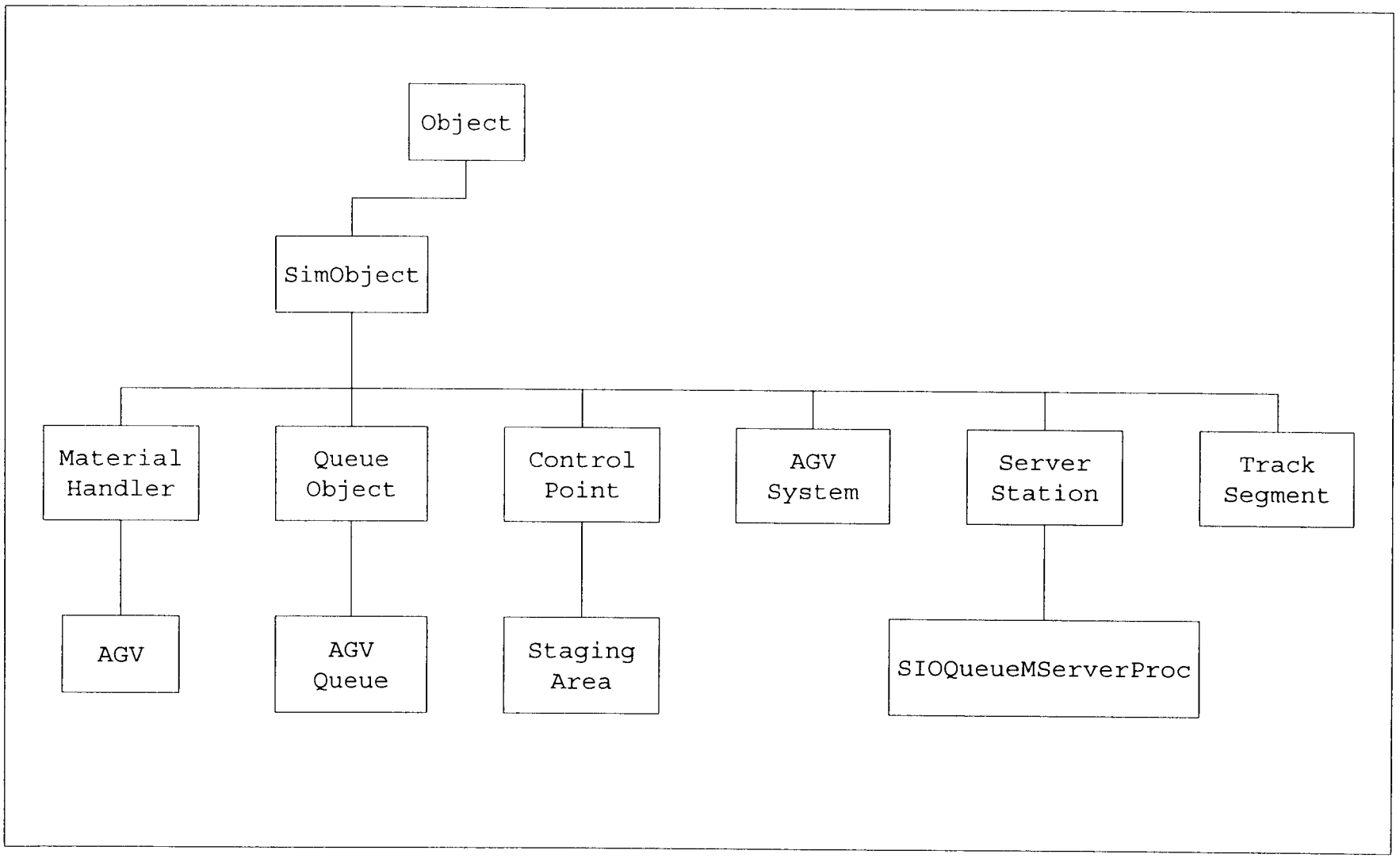


Figure 6: Extended structure of OOS for AGV Systems

1. Start an AGV's travel to a destination along a path.
2. According to its current path, check for the availability of the next control point to decide the next activity.
3. Move through the next control point without reducing its speed if there is no AGV blocking the guidepath.
4. Enter the end of a track segment and stay until the AGV that blocks its path leaves.
5. Enter a control point to perform appropriate operations.
6. Transfer parts in or out of machine cells by performing loading or unloading operations.
7. Schedule breakdown occurrences and maintenance activities.
8. Collect statistics on its operations and provide the output as requested.

Summary of the AGV Class:

Function:

To represent AGVs, especially unit load carriers, within a simulation model.

Data Storage:

- the four possible states (stop, moving empty, moving loaded and charging)
- current location in the system
- current traveling path
- the parts being carried

- reservation status by parts
- loading/unloading time duration
- the maximum speeds when empty and loaded
- acceleration and deceleration
- disposition locations when idle
- battery capacity information
- battery usage factors
- battery charging duration
- breakdown interval information and maintenance duration

Actions:

- schedule the traveling activities of AGVs
- perform loading and unloading operations
- schedule the breakdown occurrences and maintenance activities
- collect utilization and breakdown statistics information and provide output as requested

As I have mentioned, the stations in the AGV systems are machine cells. I create a new subclass called ServerStation under SimObject to define the machine cells. The ServerStation class represents the machine cells that can accept the arrival of new parts, determine an available server from among those allocated, and schedule the processing tasks. In addition, it can also trigger the activity of checking for the availability of the next machine cell, request an idle AGV, and make resource (AGV,

machine cell) reservations for parts. ServerStation class itself has two subclasses called InputStation and SIOQueueMServerProc, respectively. Those will be discussed later.

Summary of the ServerStation class:

Function:

To allow abstraction of machine cells to be defined.

Data Storage:

- name of a machine cell
- number of servers allocated
- status and statistics on each parallel server allocated
- references to the internal queues
- information of the up-stream machines

Actions:

- accept new parts
- schedule processings of parts
- transfer parts from servers to the output-queues
- trigger the activities of checking for the availability of the next machine cell and idle AGVs
- reserve machine and AGV resources for the parts in the output-queues
- release the parts that have already completed processing

- determine an appropriate up-stream machine and trigger the head part of its output-queue to move out

The two subclasses of ServerStation class, InputStation and SIOQueueMServerProc, represent the input machine cells and other regular processing machine cells of AGV systems. In the original simulation environment, after creating a new part, the Creator object passes this part to the next object (based on routing information). However, in the extended AGV system components, the new created parts are normally be directly sent to the InputStation that serves as an interfacing station between the AGV system and the other model components. The InputStation class inherits all characteristics from its superclass ServerStation, while it owns a special capability to balk parts from the system. When a part released from the Creator object arrives at the input station, it will be sent to the input-queue if no server is available to process it. However, if the input-queue is full, then this part will balk from the system. It should be noted that the InputStation class does not maintain the reservation list for newly created parts.

Summary of the InputStation class:

Function:

To represent the AGV system entry point within the simulation model.

Data Storage:

- inherits from the ServerStation class
- number of balked parts

Actions:

- inherit the actions from the ServerStation class
- collect statistics on its operations and provide output as requested

The other subclass of the ServerStation class is the class called SIOQueueMServerProc. SIOQueueMServerProc stands for single input-queue, output-queue, and multiple server processing. SIOQueueMServerProc defines the regular machine cells of AGV systems. After a part completes processing at the input station, it will pass through all needed machine cells (based on routing information) for processing until the routing is done. In addition to inheriting all properties from the ServerStation class, SIOQueueMServerProc provides the ability to maintain a reservation list for parts that are waiting in the output-queues of its up-stream machine cells. Unlike the InputStation class, SIOQueueMServerProc does not have the capability to balk parts from the system since the reservations at machine cells have been considered.

Summary of the SIOQueueMServerProc class:

Function:

To represent the single input queue, single output

queue, multiple server machine cells within a simulation model.

Data Storage:

- inherits from the ServerStation class
- reservation list for parts waiting for transport

Actions:

- inherit the actions from the ServerStation class
- update reservation information
- collect statistics on its operations and provides output as requested

Physically, the guidepath of AGV systems is comprised of control points and segments. Control points perhaps represent the loading/unloading stations, intersection points of track segments, or staging areas. These locations provide space to allow AGVs to perform appropriate operations such as loading/unloading, battery charging, and waiting. Thus, I create a class called ControlPoint to represent the control points. The ControlPoint class is a subclass of the root class SimObject. In addition, since this research only designs the full cycle technology for battery charging, staging areas offer not only space for idle AGVs, but also provide the capability to perform battery charging. Therefore, in order to distinguish between staging areas and general control points, I create a subclass called StagingArea under the class ControlPoint.

Summary of the ControlPoint class:

Function:

To represent the loading/unloading stations and intersection points of track segments within a simulation model.

Data Storage:

- name of a control point
- occupied status
- input/output-segment information
- name of an associated machine cell
- waiting list for parts being blocked in the input-segments

Actions:

- determine the shortest path from itself to another control point
- update occupied status
- trigger the next AGV waiting in a input-segment to resume its traveling
- collect utilization statistics and provide output as requested

The staging areas are the places where idle AGVs wait for the next tasks to perform. When the full-cycle charging technology is applied, these areas can also act as the charging stations. After a staging area becomes idle, it will automatically search for AGVs whose batteries are

nearly depleted and then trigger one of them to come for battery charging. For control purposes, reservations at staging areas for either waiting or charging are necessary.

Summary of the StagingArea class:

Function:

To represent the staging areas within a simulation model.

Data Storage:

- inherits from the ControlPoint class
- reservation information by AGVs

Actions:

- inherit the actions from the ControlPoint class
- After releasing an AGV, trigger other AGVs that require to move to staging areas to travel
- schedule the battery charging operation
- update the reservation information for AGVs

In addition to control points, the other component of AGV guidepath is segments. Segments represent the guidepath between control points. Each segment has its own length, capacity for AGVs, and direction capability. In this research, the capacity of each segment for AGVs remains at one. In addition, segments are restricted to be unidirectional. I add the TrackSegment class under the root class SimObject to represent segments.

Summary of the TrackSegment class:

Function:

To represent AGV guidepath segments within a simulation model

Data Storage:

- name of a segment
- begin and end control points of a segment
- occupation status by AGVs

Actions:

- trigger an AGV waiting at the begin control point to start its traveling
- update the occupation status
- determine the shortest path from its end control point to a control point
- collect utilization statistics and provide the output as requested

In the original environment, there was a simulation processing class called Queue used to represent the queue objects of machine cells. This class is defined with the procedures to store other objects within an ordered linked list, to remove objects from the front of the queue, to search the queue for specific objects, and to collect and output statistics on its activities. As I have assumed before, the queue objects of each machine cell of the AGV systems have limited sizes and are associated with loading/unloading stations. Thus, due to these adjustments,

I create a subclass called AGVMachQueue under the existing Queue class.

Function:

To provide queue building blocks for the construction of a machine cell.

Data Storage:

- inherits from the Queue class such as the first and last parts in the queue, current total number of parts, and the time a part has spent in the queue
- size limit of a queue
- control point reference

Actions:

- Addition and removal of objects to/from the queue
- make sure there is space available upon request by parts
- collect queue size and part queuing time statistics and provide the output as requested

Another simulation processing object needed is the class called AGVSystem. The AGVSystem class provides several capabilities which include the ability to:

1. Select an AGV in accordance with the AGV selection rule.
2. Find a part waiting for transport in accordance with the AGV dispatching rule.

3. Select a specific AGV that has no tasks to perform, no staging area to go, and is in stop status.
4. Find an AGV that is required to perform battery charging.
5. Search for an AGV with a specific current location
6. Remove an AGV from the AGV fleet list

The information maintained in the AGVSystem class includes AGV fleet, control points, segments, and machine cells. The AGVSystem class provides a data base through which all requests from other system objects can be responded and acts as a controller of AGV system component interaction for the entire model run.

Summary of the AGVSystem class:

Function:

To maintain the information of AGV system components. Act as a controller to respond to requests from other system objects.

Data Storage:

- pointers to the AGVs in the system
- pointers to the machine cells in the system
- pointers to the control points in the system
- pointers to the track segments in the system
- AGV dispatching rule information
- AGV selection rule information

Actions:

- respond to the requests from other objects
- maintain system component information

3.3.2 Conceptual Design of AGV Systems

Structure of Part Flow in AGV Systems

Before I describe the detailed conceptual design of AGV systems, it is better that we understand the general job flow of AGV systems. The job flow diagram is shown in Figure 7. This diagram is broken into three major parts: Activities of InputStation, AGV Behaviors, and Activities of SIOQueueMServerProc Station. In this section, I will use this structure as a guide to systematically describe the detailed conceptual design of AGV systems.

Activities of InputStation

Recall the general AGV system layout shown in Figure 1. Parts arrive to the system through the input station. Typically, a part needs to be loaded individually onto a fixture at the input station before it can be machined at any of the other processing machine cells. We may consider that the input station has an input-queue, numerous servers, and an output-queue. Input-queue and output-queue offer space to temporarily store new arriving parts and fixtured parts, respectively. The servers are responsible to perform

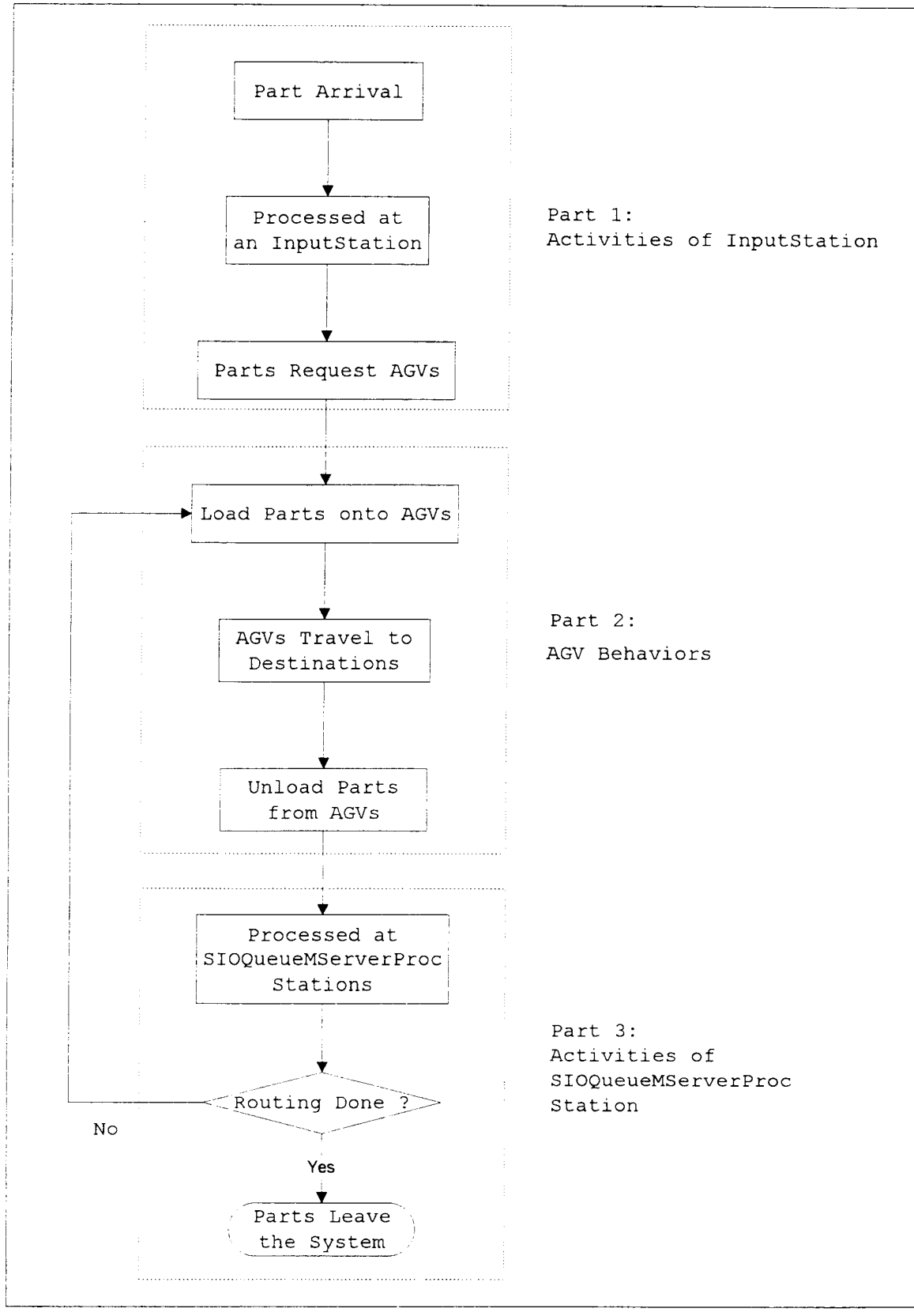


Figure 7: Job flow through an AGV system

fixture processings. When a part arrives at the input station, it first checks for the availability of the servers. If there is a server available, this part will enter the available server to be processed. But, if all servers are busy, we may use the input-queue as a buffer. If the input-queue has enough space, the part goes to an appropriate position and waits; otherwise, this part balks from the system.

After the process is done, the processed part is ready to depart the server. There are two situations that could happen (refer to Figure 8):

1. If the output-queue has enough space, place the part onto the output-queue, and then remove another part from the input-queue and begin its processing.
2. If the output-queue is full, the processed part keeps blocking the server until the output-queue is available.

Once a part departs the server, it goes to the output-queue to wait for transport to the next machine cell. Thus, the next thing concerned is how a part can request an idle AGV for transport to the next machine cell. The detailed design to accomplish this is shown in Figure 9.

When a part enters the output-queue and hits the end of the queue, it can start to check for the availability of the next machine cell in accordance with its processing route. If it is not the first part (head part) of the queue, it must wait. The head part will make sure whether the next

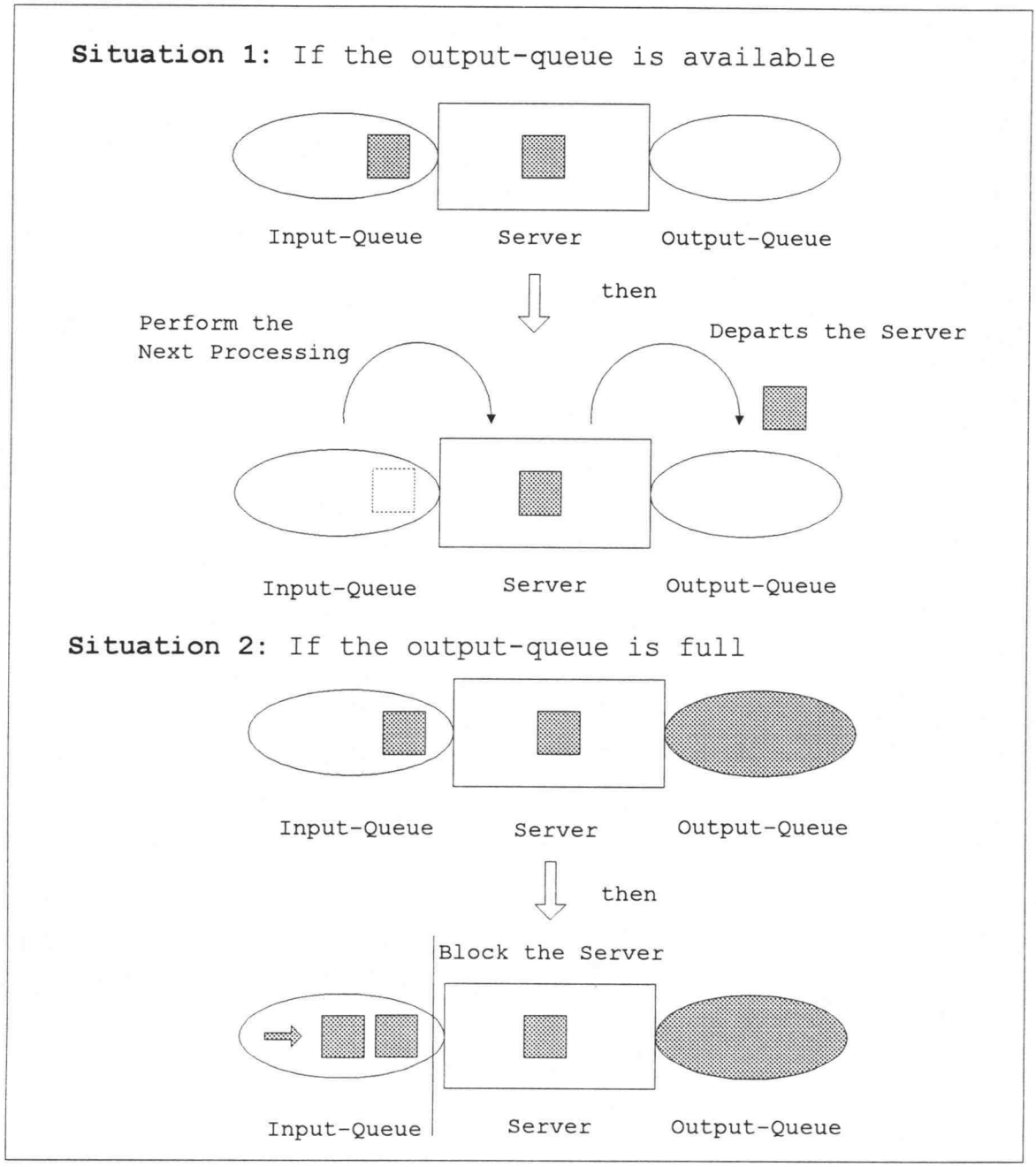


Figure 8: Possible queue manipulations upon completion of part processing

machine cell is available by checking its servers and input-queue space status before it can request AGV service. If the next machine cell has available space either from the

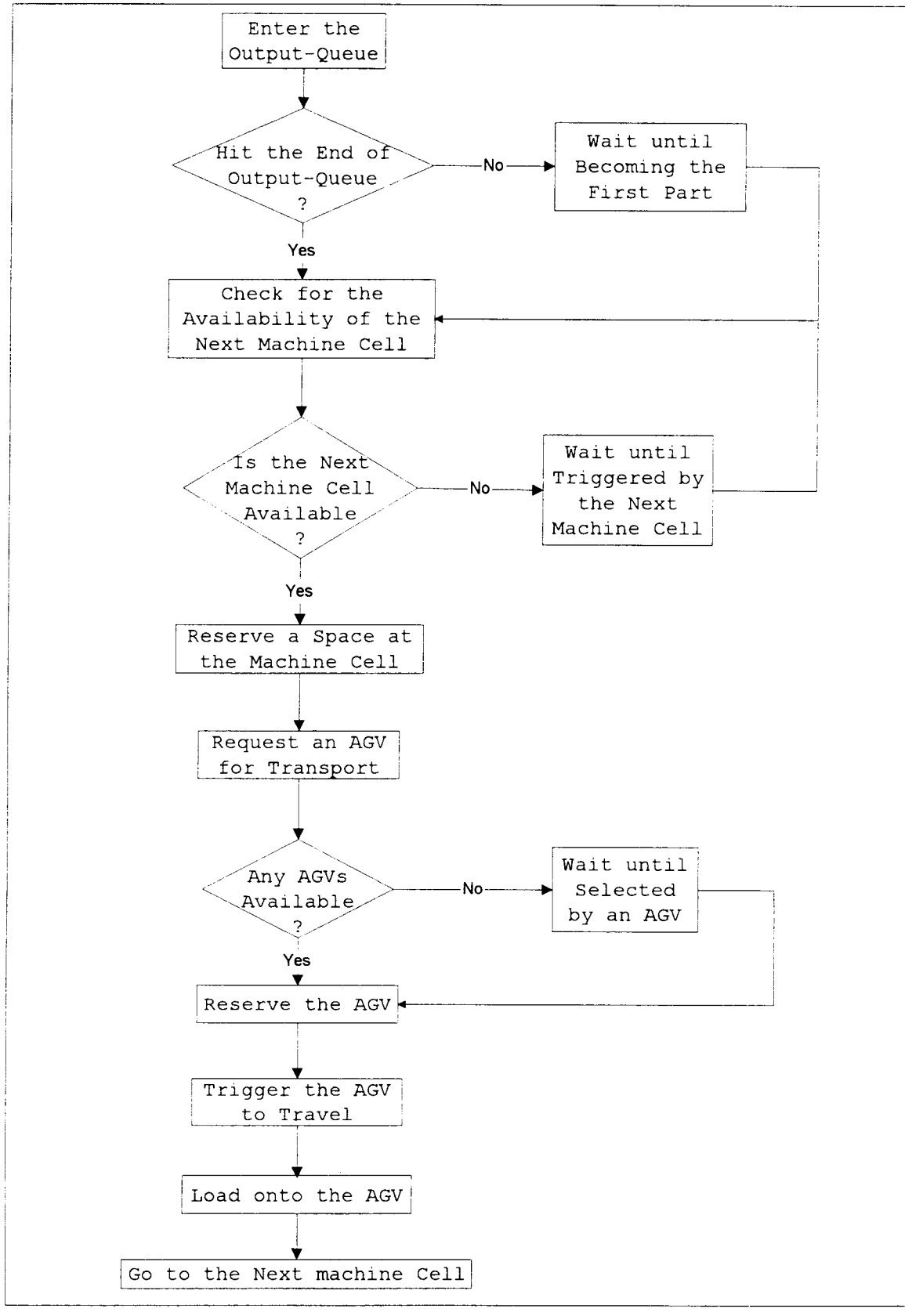


Figure 9: Part activities in the output-queues

servers or the input-queue, the head-part needs to make a reservation at that machine cell immediately. The reservation is only for a space rather than for a specific server or position in the input-queue. However, if the next machine cell is not available to accept the part, this part has to stay at the current position to wait for the trigger by a machine cell.

Once the part reserves a space at the next machine cell, it obtains the right to find an idle AGV and request it for transport. The rule regarding how to select an AGV is called AGV selection rule. In this research, I only design two options: "LowestUtilization" and "Nearest". If the part finds an AGV, in order to avoid that this idle AGV will be requested by another part at other machine cells, it must reserve this AGV immediately. But, if no AGVs can be requested, this part needs to stay in the current position to wait for future selection by another idle AGV.

After the part reserves an AGV, it may trigger that AGV to start traveling from its current location to the output-queue where the part is located. When this reserved AGV arrives at the loading station associated with the output-queue where the requesting part is located, four things should be carried out (refer to Figure 10):

1. Load part A onto the AGV.
2. Trigger the requesting part's successor (part B) to start moving out of the output-queue by doing the same activities that have been described previously.

3. If there are parts blocking the servers, remove one of them (part D) from a server in accordance with the "First Block-First Out" rule and place it onto the output-queue. This will automatically trigger the activity which I have described in the case 1 in Figure 8.
4. Send part E to the server and perform its processing.

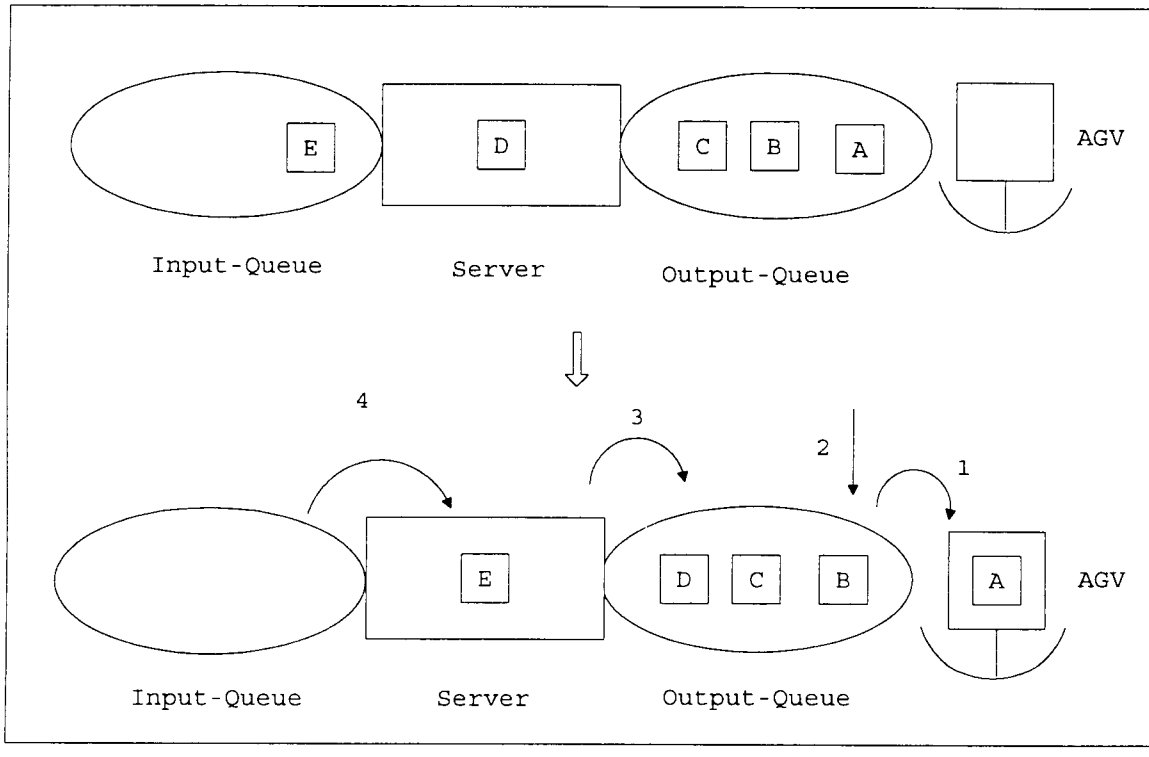


Figure 10: Activity when an AGV arrives at a requesting station

AGV Behaviors

The behavior of an AGV is closely linked with other components in the manufacturing system. The general behavior architecture is shown in Figure 11.

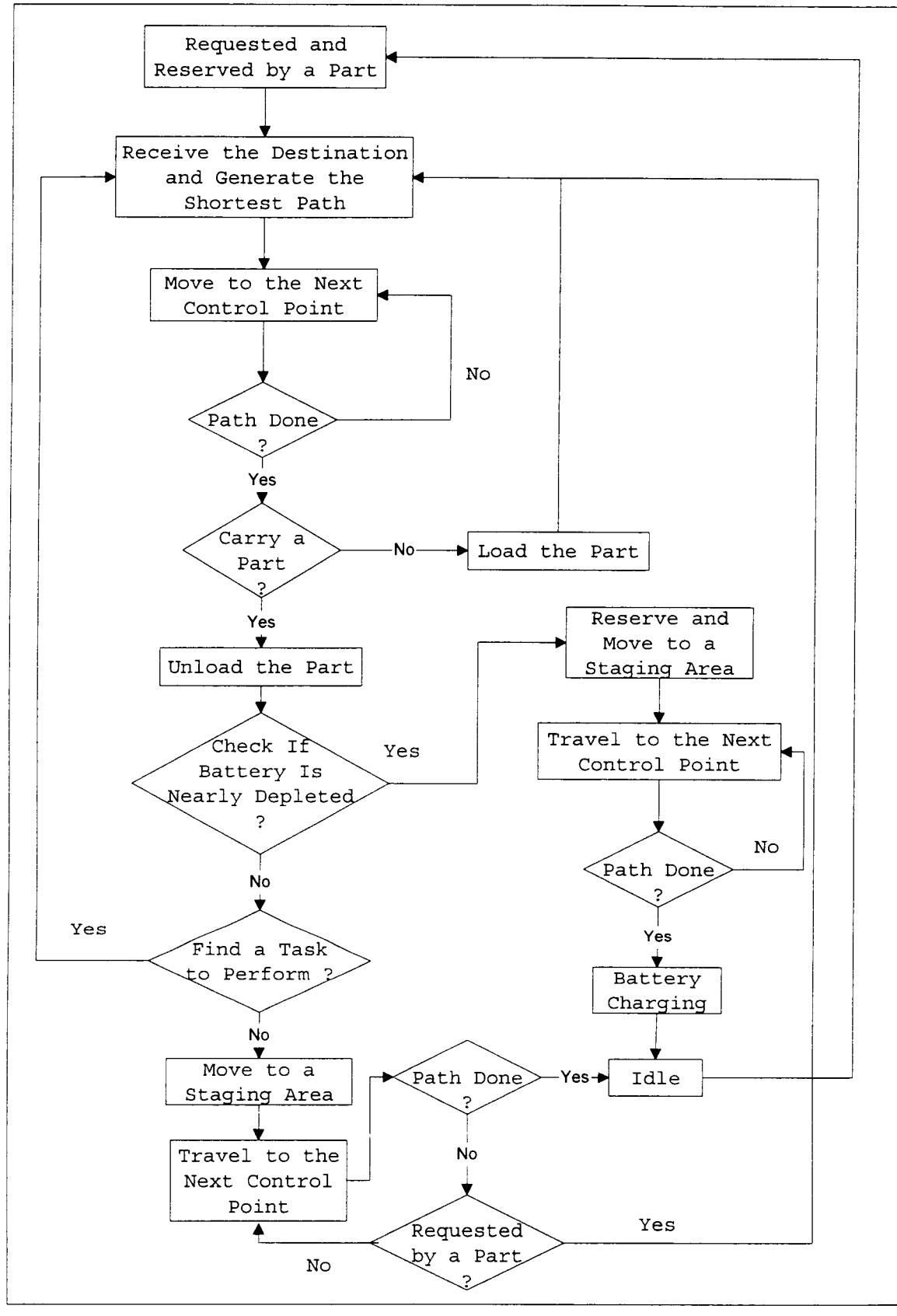


Figure 11: General behaviors of AGVs

- Requested and reserved by a part:

At the beginning of simulation, the initial locations of AGVs can be anywhere on the guidepath in the system. These places could be the staging areas, control points, or ends of track segments. The movements of AGVs are triggered by the requests from the parts that are waiting in the output-queues of server stations for transport. As I have mentioned before, when a part finds an idle AGV, it will immediately reserve it, while this reserved AGV will also reserve that part at the same time. This mutual reservation avoids AGV/part reassignment until service is completed. But, if the AGVs are traveling to staging areas and are requested, it will discard the current task (go idle at staging area) immediately. In addition, they need to cancel the reservations at the staging areas to which they were traveling.

- Receive the destination and generate a path:

As soon as a part reserves an AGV, it may trigger the AGV to travel by sending the information of current location that can be used as the destination for the AGV to generate a path. This path is assumed to be the shortest path from the AGV's current location to the part's location. The general idea for the AGV to generate the shortest path is that according to the overall layout information from the system controller (AGVSystem object), the AGV is able to examine all possible paths (comprised with control points)

that can get to the destination until the shortest one is found. To accomplish this, I develop a best-first search technique that will be discussed in detail in section 3.4.

- Move to the loading/unloading station:

After generating the shortest path, the AGV starts traveling. Figure 12 shows the anatomy of AGV movements along the guidepath. To avoid collision, when the AGV at control point 1 intends to start traveling, it needs to make sure whether there is another AGV occupying segment 1 first. If so, it has to stay at the current location and wait until that AGV leaves segment 1. However, if segment 1 is not occupied, the AGV is allowed to depart control point 1.

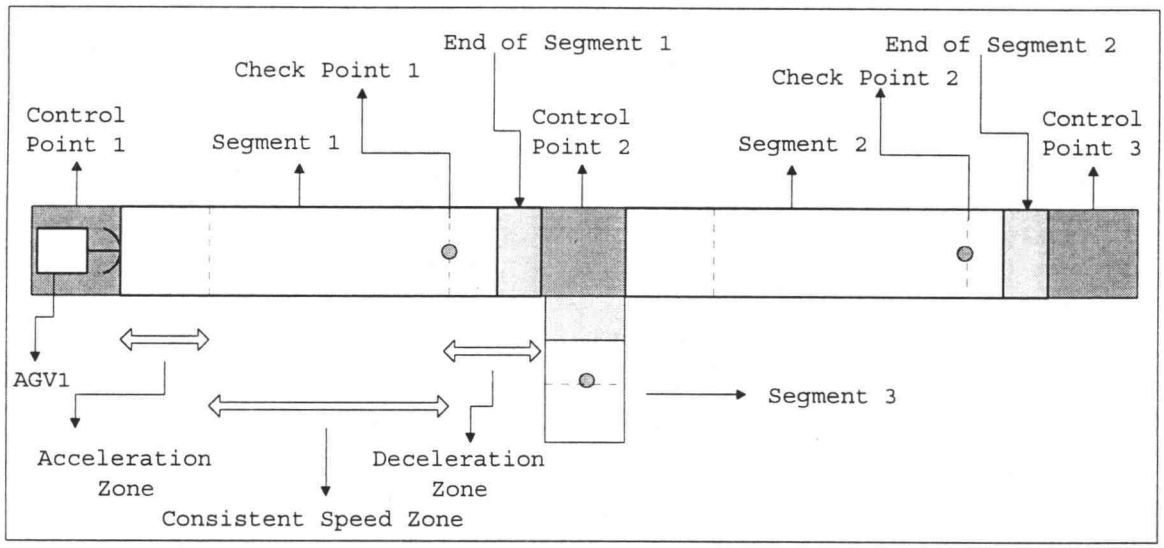


Figure 12: AGV movements along the guidepath

When the AGV travels along a segment, it may experience three different states: accelerate, move at a consistent speed, and decelerate. After the AGV starts moving from static status, it has an acceleration period to increase its speed from zero to a consistent status. Then, the AGV moves over a period of time using a consistent speed until it arrives at a place called check point. The check point that is not really a physical point is used to trigger the AGV to check for the availability of the next control point before it continues its traveling. More specifically, when the AGV 1 arrives at check point 1, it automatically checks if control point 2 is occupied by another AGV. If control point 2 is occupied, then AGV 1 has to reduce its speed to be ready to enter the end of segment 1. The end of segment is assumed to be the place where a blocked AGV waits to resume its travel. But, if control point 2 is not occupied, AGV 1 then needs to check if control point 2 is the destination. If so, AGV 1 has to reduce its speed to enter control point 2 to perform loading/unloading operations, battery charging, or just wait for future tasks. However, if control point 2 is not the destination, AGV 1's movement will depend upon the occupation status of segment 2 and 3. If segment 2 is occupied by another AGV, the AGV will reduce its speed to enter control point 2. If segment 2 is not occupied, AGV 1 needs to check for the status of segment 3. This situation is especially important for a control point (control point 2) that has multiple input-segments. If

there is another AGV traveling on segment 3 and very close to control point 2, then AGV 1 must enter the end of segment 1 to avoid collision. If segments 2 and 3 are not occupied by any other AGV, AGV 1 may move through control point 2 without reducing its speed. Once the AGV moves through control point 2 without reducing its speed, it will keep moving at its current speed until it arrives at the next check point (check point 2). When arriving at check point 2, it will do the same decision procedures as it did at check point 1. To sum up, the above decision procedures are shown in Figure 13.

- Determination of moving speed:

Refer to Figure 12, I call the areas where the AGV moves with acceleration, consistent speed, or deceleration the acceleration zone, consistent speed zone, and deceleration zone, respectively. The determination of AGV's moving speed is considerably related to these zones. When an AGV starts moving, it should have already determined what speed it may use to pass this segment. First, it has to determine the required lengths of both acceleration and deceleration zones by initially using its maximum speed as the consistent speed. An AGV has two different maximum speeds associated with whether it is loaded or not. Secondly, then, subtract the sum of these two zones' lengths from the length of the segment to which it is moving.

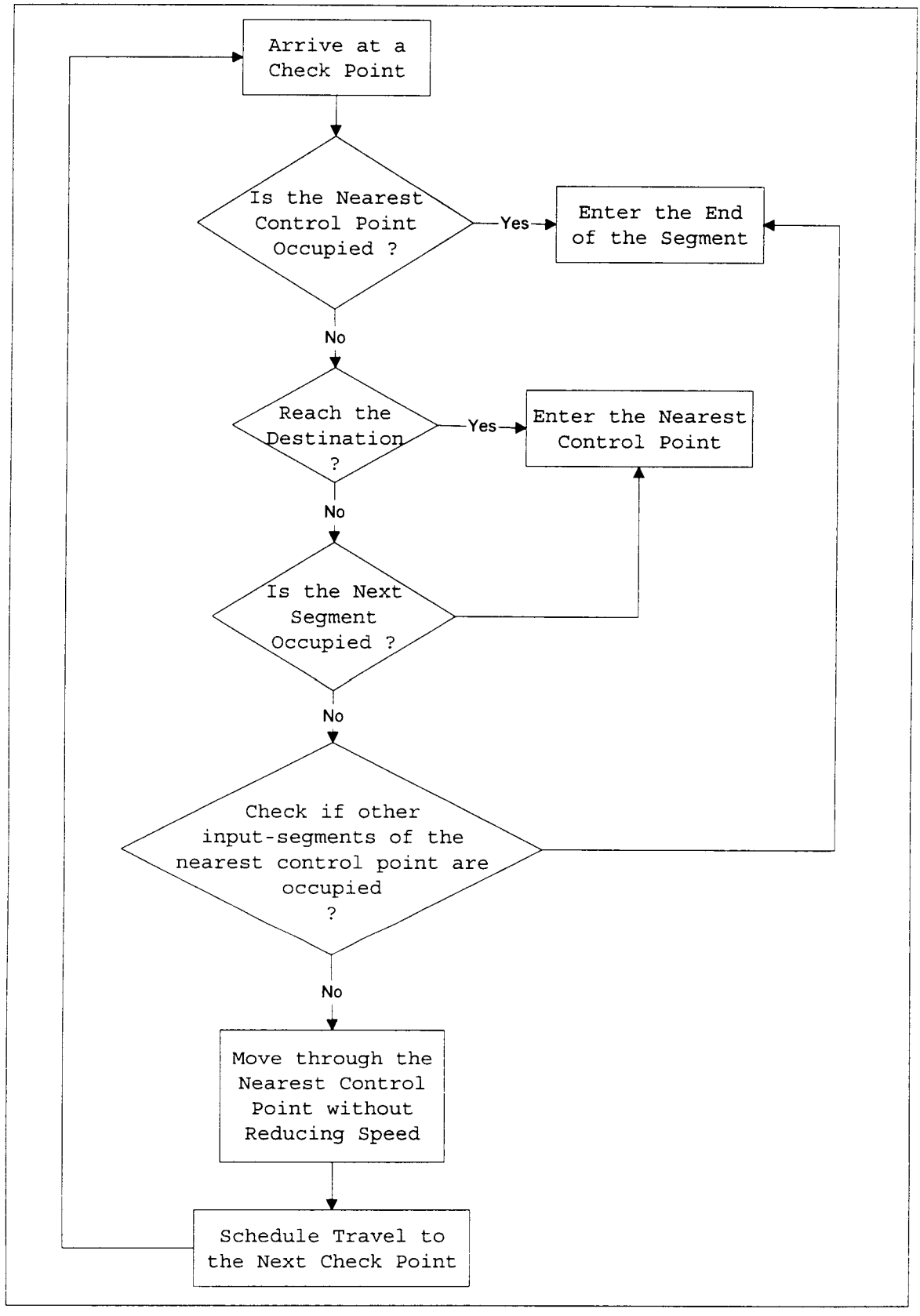


Figure 13: Decision procedures for AGV movements

If the result is positive, that means the AGV is allowed to move through this segment using its maximum speed. However, if the result is negative, the maximum speed can not be used. Thus, the next question is what speed the AGV can use. To deal with this situation, I simply assume that the AGV can only use 50% of its maximum speed as the consistent speed during its traveling on the consistent speed zone. Since I don't consider the turning situations in the simulation, the AGV is assumed not to reduce its speed when encountering a curve.

- Traffic congestion:

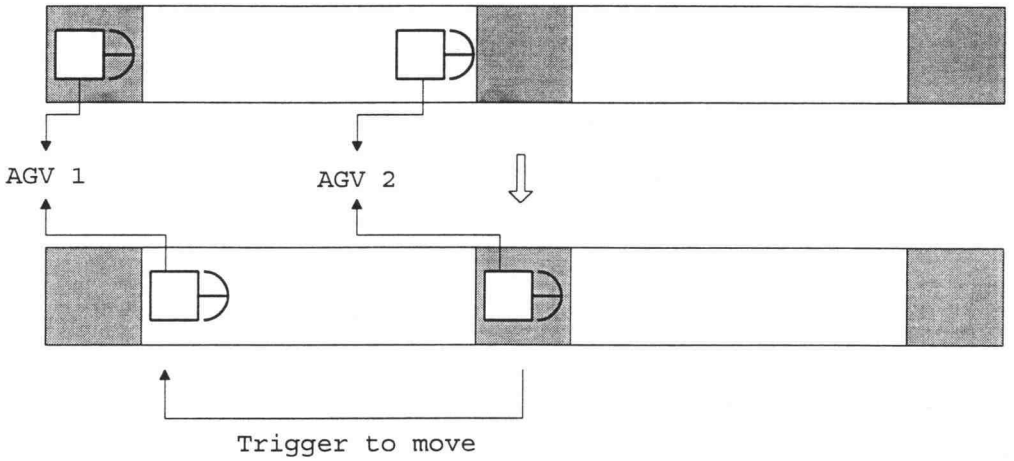
There are two situations when traffic congestion occurs. These two situations are shown in Figure 14.

Case 1: When an AGV at a control point is blocked by another AGV that is either traveling along or waiting in the output-Queue, it can not move until that AGV leaves the segment. As soon as that AGV leaves the segment, it automatically triggers the blocked AGV to start moving.

Case 2: An AGV is blocked at the end of the segment, if there is another AGV in the end-point of the segment at which it is locating. Refer to Figure 14 again, both AGV 1 and 3 are blocked by AGV 2. When AGV 2 leaves the control point, it will trigger either AGV 1 or AGV 3 in accordance with what rule is used. In this research, I have previously assumed to use FIFO rule (assumption 15).

Case 1: an AGV is blocked at a control point:

After AGV 2 leaves the segment, it triggers AGV 1 to move.



Case 2: an AGV is blocked at the end of a segment:

After AGV 2 leaves the control point, it triggers AGV 1 to move because AGV 1 has a longer waiting time.

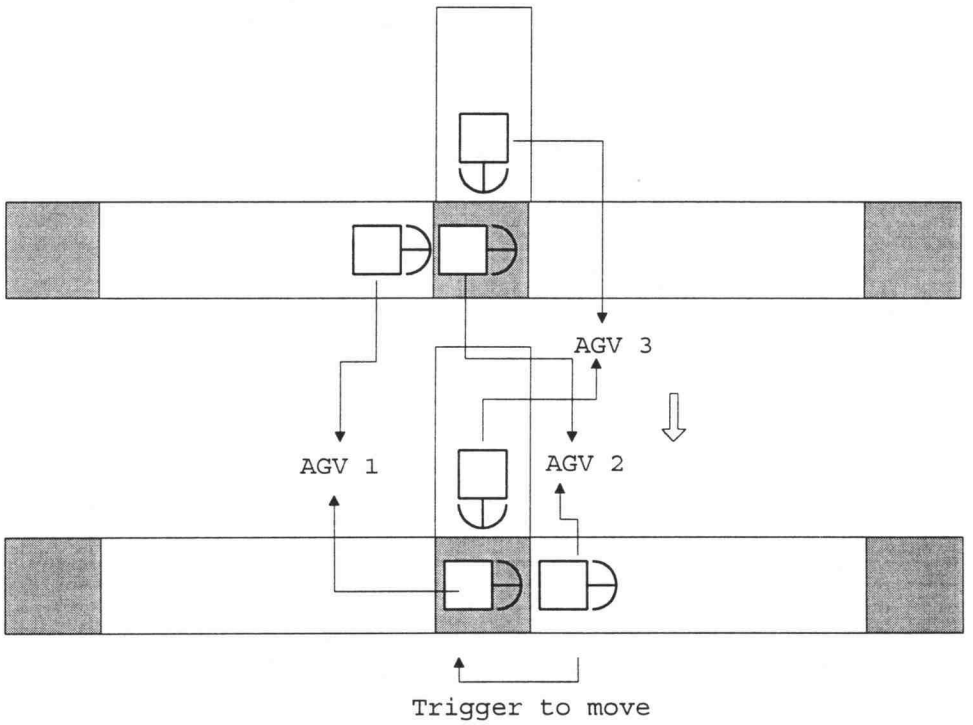


Figure 14: Two traffic congestion situations

- Perform loading/unloading operations:

When an AGV finds out that its path is done at the check point, it will automatically enter the control point connecting to the segment at which it is traveling. After entering the destination, it could perform loading or unloading operations in accordance with whether it is loaded or not. If it is loaded, the AGV will unload the part being carried to the machine cell from the work platform by using its automatic loading/unloading devices. If unloaded, it will perform the loading operation to put a part on its work platform. Once the AGV completes the loading/unloading operations, the AGV's continuous movements become much more complicated (refer to Figure 11). If the AGV completes the loading task, it will obtain the information of the next destination from the part it loads and generate the shortest path to move there. However, if the AGV completes the unloading operation, several things need to be considered.

First of all, the AGV has to check if its battery is nearly depleted. The AGV is designed to check its battery capacity only after it completes the unloading task during the simulation. I have assumed that if an AGV's current battery capacity is lower than 20% of its maximum capacity, it needs to go to the nearest staging area to charge its battery. Once the AGV finds the appropriate staging area, a mutual reservation between the AGV and the staging area is required. This mutual reservation is critical because it

prevents the AGV from being dispatched to another task when it is on the way to the staging area as well as the staging area being claimed by another AGV.

If the AGV does not require battery charging, it becomes idle and can be dispatched to deliver another part waiting for transport. Selecting an appropriate part is done by using the dispatching rule (refer to section 2.2). In this research, I only design two rules: "Nearest" and "EarliestQueueArrivalTime". But, if the AGV can not find any part to deliver, that is, there is no part waiting for transport in the system at this time, it has to move to the nearest staging area which is not occupied or reserved by any AGVs. Unlike the battery depleted case, when an idle AGV finds an appropriate staging area, the mutual reservation is not necessary. That is, the selected staging area will not admit another AGV (depleted battery or idle), but the idle AGV can be requested by any parts which need to be delivered. Therefore, if an AGV on the way to a staging area is requested by a part in certain machine cell's output-queue, it must immediately go to the new destination to pick up that part instead of traveling to the staging area.

- Idle and battery depleted:

There are two situations under which AGVs will move to staging areas: idle or depleted battery. The charged AGV will become idle after the charging operation is over. The

idle AGVs keep waiting in the staging areas until they are requested by parts. Once an AGV leaves the staging area, the staging area becomes available. This available staging area then immediately checks if there are any AGVs waiting for battery charging and, if present, triggers one to come. Meanwhile, if there is more than one candidate waiting, the staging area will select the one with the lowest battery capacity. These candidates must be the AGVs with nearly depleted batteries that are either stopped or traveling to a loading station. On the other hand, the loaded AGV with a fully depleted battery that is traveling to an unloading station will not be selected. But, if there is no AGV waiting for charging, the available staging area will next check if there is any idle AGV waiting for a staging area. If there is more than one idle AGV, the AGV with the nearest distance to the staging area will be selected.

A fully depleted battery is a battery whose capacity becomes zero. To calculate battery consumptions, several battery usage factors are used. These factors are stated in ampere-seconds (A-sec). A summary of battery usage factors follows:

- moving empty and moving loaded factors (A-sec/ft)
- accleration and deceleration factors (A-sec/each)
- loading and unloading factors (A-sec/each)

An AGV's battery capacity is stated in amphere-hours (A-hour). Once AGVs consume their battery capacity due to

traveling, loading or unloading, we must adjust its battery capacity by using appropriate factors. For example, a static AGV is assumed to have a 100 A-hours capacity and have following usage factors:

- moving empty factor (5 A-secs/ft)
- moving loaded factor (15 A-secs/ft)
- accleration factor (10 A-secs/each)
- deceleration factor (3 A-secs/each)
- loading and unloading factors (20 A-secs/each)

If it needs to travel 20 ft to a loading station to load a part, the required battery consumption is:
[10 A-secs/each * 1 (accleration)] + [5 A-secs/ft * 20 ft (moving empty)] + [3 A-secs/each * 1 (deceleration)] + [20 A-secs/each * 1 (loading)] = 133 A-secs

Thus, after performing these tasks, this AGV's battery capacity remains:
100 - 133/3600 = 99.963 A-hours (1 A-hour = 3600 A-secs)

- Breakdowns:

In this research, AGV breakdowns can be classified into two types: general and battery thoroughly depleted. The first type are due to human errors, track failure, mechanical trouble, etc. The second type result from AGV batteries which become fully depleted. In both cases, once a breakdown occurs, the failed AGV will immediately stop

performing its task and be taken off the guidepath to repair. During the repair duration, this AGV can not be requested by any parts or dispatched to any tasks. When the repair is done, this AGV will be put back to the location where the breakdown occurred and resume its unfinished actions.

Parts in SIOQueueMServerProc Stations

When a part is unloaded from an AGV, it enters the regular processing machine cell. If there is a server available, it goes to the server to be processed; otherwise, it goes to an appropriate position in the input-queue. Since this part has already reserved a space before coming to this machine cell, space availability at this point is not a concern. After the processing is done, the part is sent to the output-queue and waits for transport by an AGV to the next machine in accordance with the processing route. All required operations regarding these internal part actions are similar to those I have described previously. I mentioned before that if an output-queue head part can not reserve a space at the next machine, it has to wait for the trigger from a down-stream machine cell. The detailed design of this is shown in Figure 15.

The machine cell 1 is the up-stream machine cell of machine cell 2. When an empty AGV arrives at the machine cell 2, it loads part 1 and then moves to the next station. The machine cell 2's internal part actions are same as those

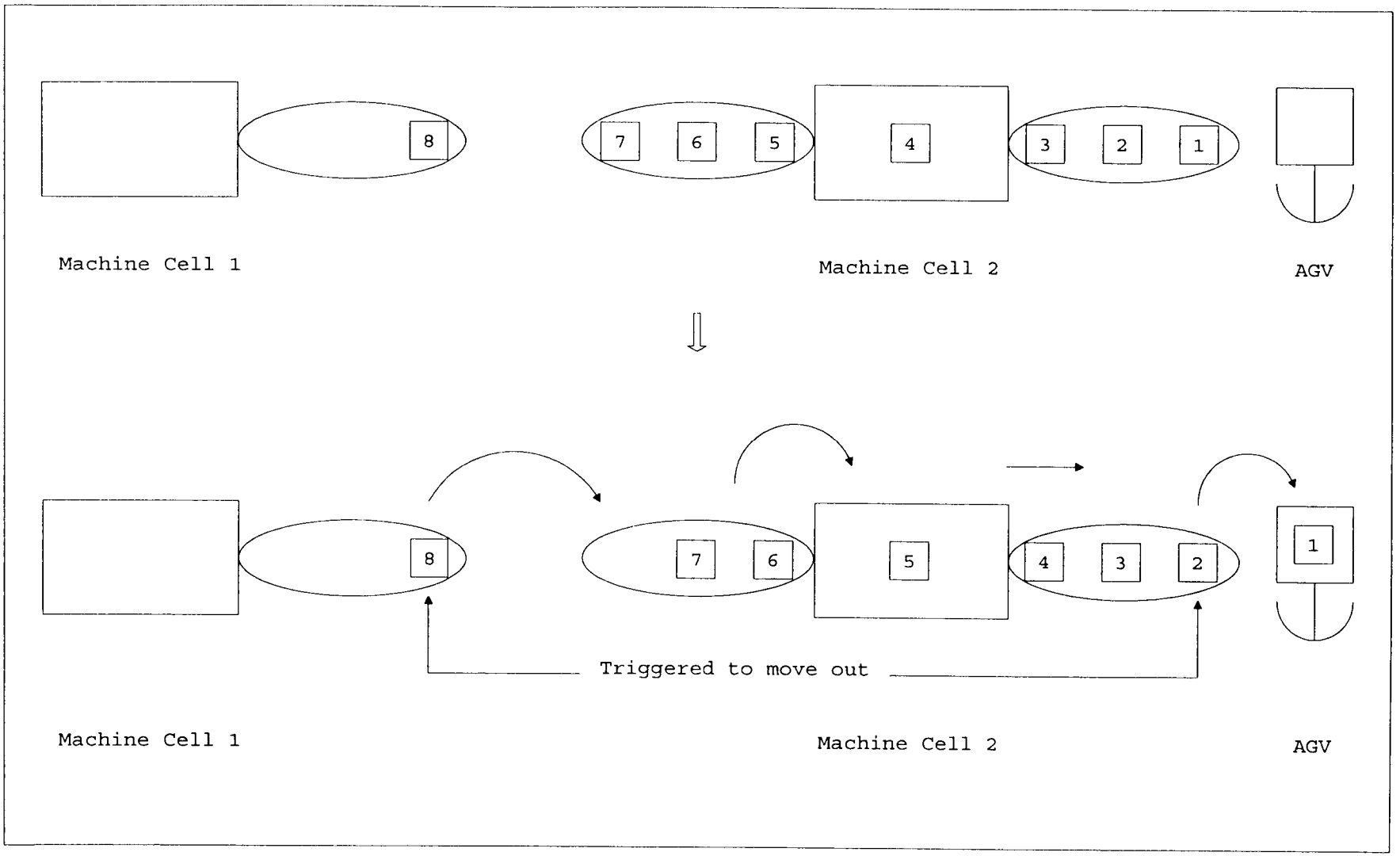


Figure 15: Architecture when loading parts onto AGVs

shown in Figure 10. In addition to these internal actions, machine cell 2 may trigger the part (part 8) waiting in its up-stream machine cell to start moving out because machine cell 2 has a space available now. Since machine cell 2 might have more than one up-stream machine cell (depending on parts' individual routings), a decision rule to determine an appropriate part to move out is required. In this research, the default rule is the "Longest output-Queue Waiting Time". That is, machine cell 2 has to determine which output-queue's head part of up-stream machine cell has the longest queue waiting time and then triggers this part to start moving out by requesting an idle AGV.

3.4 Implementation of Object-Oriented AGV System Simulation in Smalltalk

As mentioned before, to simulate an AGV system using a traditional simulation paradigm is still a painful process at this time. However, object-oriented simulation can significantly remedy this problem. In an object-oriented environment, objects can be created easily and the detailed internal actions of the real-world objects are hidden by one of OOP's basic characteristics, encapsulation. This simplifies the modeling of a manufacturing system with AGV material handling. This research uses the techniques of discrete event simulation to design the simulation model of

AGV systems. Through event scheduling, method execution and message passing between different objects, the simulation environment can be easily built.

- Machine Cells:

To implement the simulation of machine cells in AGV systems, I add ServerStation, InputStation, and SIOQueueMServerPro classes into the original environment. ServerStation is under the SimObject class. InputStation and SIOQueueMServerProc are the subclasses of ServerStation. The implemented ServerStation class provides general functionalities for the simulation of machine cells, including the processing of parts, part departure from server, part departure from output-queue, checking for the availability of the next machine cell, checking of AGV availability, reservation, and AGV movement triggering. The InputStation and SIOQueueMServerProc classes inherit these functionalities, while they have their own ones such as entry of parts, etc.

The ServerStation class has inputQueue, outputQueue, and numberOfServers variables to represent the physical structure of a machine cell. Since parts arrive to the system only through the input station, after a part is generated by the Creator object, it will be sent to the InputStation by scheduling a partArrival event. Once it goes to a server processing, the InputStation will schedule a partProcessDone event through which it can trigger this

part to leave the server if the output-queue is available. If the output-queue has space available, the partDepartureFromServer method is executed to remove a part from the server. That may trigger it to start moving out of the input cell by checking for the availability of the next machine cell. In addition, it can also remove another part from the input-queue to set up its processing. If the part becomes the first part of the output-queue, it can check for the availability of the next machine cell in accordance with its processing route and make reservation if the next machine cell has space available. These are done by the execution of checkNextMachAvailability and reserveMachSpace methods. After the part makes the reservation, it requests an AGV for transport to the next machine cell. Two methods are implemented to accomplish these actions, checkAGVAvailability and reserveAGV. They are triggered by previously mentioned checkNextMachAvailability method. As soon as the head part finds and reserves an idle AGV, it may trigger the AGV to start traveling by sending it the information of the shortest path which is determined through the communication with the AGVSystem object. The method triggering AGVs to move is the method called triggerAGVTravel that will schedule a traveling event for the AGV being triggered.

- Control Points, Staging Areas, and Segments:

  These are the basic components of the AGV guidepath.

To represent these physical objects, I add the ControlPoint and TrackSegment classes under the SimObject class, and the StagingArea class under the ControlPoint class. The ControlPoint and TrackSegment classes provide several similar functionalities such as generation of the shortest path, requesting of the next AGV, and maintain occupation status.

Consider, a control point of AGV system may have several input and output segments but a segment can only have one begin point and one end point. Thus, through the linkage between control points and segments, the whole AGV guidepath can be constructed. Both ControlPoint and TrackSegment classes have a status variable that indicates whether they are occupied by an AGV. I use 0 and 1 to define the empty and occupied status, respectively. Thus, when AGVs enter or leave, the status needs to be updated.

For control purposes, once an AGV leaves a control point or segment (status value changes to 0), the requestNextAGV method will be executed to schedule a travelAlongPath event for a blocked AGV. For control points, this may trigger one of the AGVs blocked at the ends of the inputSegments to start moving. The determination of an appropriate AGV among those is based on the "First Come-First Out" rule. For segments, this can only trigger a blocked AGV to start moving from the unique beginPoint.

The most critical functionality both ControlPoint and TrackSegment provide is the generation of the shortest path

between two control points or between a control point and a segment. I apply a best-first search to deal with this determination. The general idea for the AGV to generate the shortest path is that, according to the overall layout information from the system controller (AGVSystem object), I examine all possible paths (comprised with control points) that can get to the destination until the shortest one is found. The algorithm of this best-first search includes five steps:

1. remove the best path from list (openPaths)
2. test for goal, if found, then stop
3. generate child paths
4. order list by sorting on path distance and with shortest path first
5. return to step 1

Let us suppose that we want to find the shortest path from point 1 to point 2. Point 1 is the first state (start point) and point 2 is the goal. First, we test point 1 for goal, if not, then generate all children of point 1 to create new possible paths by linking the first state and child states. This can be done by using the linkage information of guidepath I mentioned previously. Next, put all newly created paths at the end of the openPath variable. This openPath variable is a SortedCollection object that stores all path lists comprising control points. Each path stored in the openPath variable is designed to be an ordered

collection object with two elements. The first element is the length of this path. The second element is the path which is an ordered collection object of control points comprising the path. When storing these paths, the openPath variable can sort on the lengths of all paths. Thus, in this way, the first path in the openPath variable can be guaranteed to be the current shortest one. In step 2, we check if the first state of the first path is the goal. If so, we stop searching and this path is the answer; otherwise, we continue to generate all children of the first state and examine the best path until we find the shortest path. This algorithm is guaranteed to find the shortest path without requiring to generate all possible paths. To avoid cycling search, I assumed that the size of any path must be less than the sum of total size of control points (including staging areas) in the system plus two. In other words, once a generated path with a size which is equal to this limited value (for example, if the total points in the system is 10, this limited value is 12), it needs to be discarded and can not be put in the openPath variable. Since I have assumed that the length of control points is zero, the determination of the shortest path between a control point and a segment is similar to the above algorithm. Supposing that we want to generate the shortest path from segment 1 to control point 1, what actually need to do is to find the shortest path from the endPoint of segment 1 to control point 1.

AGVs:

To implement the simulation of AGVs, I add the AGV class under the MaterialHandler class. Since each AGV is only in the following status during operations: stop (static or still perform loading/unloading operations), moving with loads, moving without loads, or charging, I implement four status variables to represent AGV's status which are stop, movingEmpty, movingLoaded, and charging. These status variables will be synchronously updated in accordance with the change of AGV's tasks during simulation.

Once an AGV is requested by a part in the system, it will be triggered to move to the destination by executing the travelAlongPath event that is scheduled by previous execution of the triggerAGVTravel method of ServerStation. When an AGV starts moving from stopStatus, it needs to appropriately update its status variables according to its load status. For example, if it is not loaded, it should change the stopStatus value from 1 to 0 and change the movingEmptyStatus value from 0 to 1. In addition, during simulation the AGV needs to update its currentLocation and path variables. The path variable storing the sequential control points along the path is defined as a collection structure.

The AGV movements are controlled by scheduling appropriate movement events according to its current location, status, and path completion status. As triggered by a part, if its current location is a control point or

staging area, the next movement event will be scheduled in accordance with its path size. If its path size is greater than 1, the checkNextPoint event, which triggers the AGV to check the occupation status of the next control point, is scheduled. If its path size is 1, that is, the current location is the destination, the enterControlPoint event is scheduled.

But, if its current location is a segment, the scheduling of movement events is much more complicated. If the endPoint of the segment at which it is locating is available and its status is stop, I schedule the enterPoint event to trigger the AGV to start entering the control point. If its status is moving (either empty or loaded), two situations may occur (Figure 16). If the AGV's location is between point 1 and check point, we don't schedule a new event because the AGV's current task is going to the check point. However, if its location is between the check point and point 2, we need to schedule the enterPoint event to push the AGV to enter point 2 rather than allowing the AGV to perform its scheduled task.

When the AGV arrives at a check point, the checkNextPoint event is executed. That will trigger the AGV to check if the next control point and the output-segment of the next control point are available. In addition, it will also trigger the AGV to check if the destination is reached.

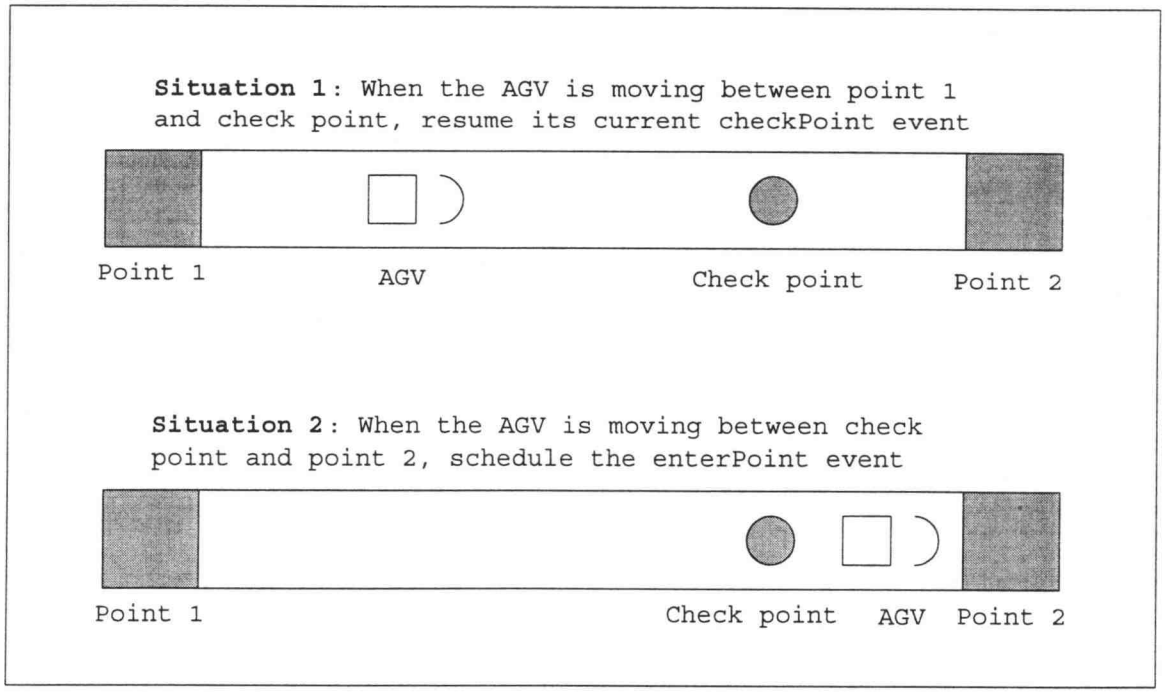


Figure 16: Two possible events occur if the requested AGV is moving on the segment

If the next control point is occupied, the researchEndOfSegment event is scheduled. If not, but the next point is the destination, the enterPoint event is scheduled. However, if the next control point is not occupied, we need to check the status of the output-segment of the next point. If that segment is occupied, the enterPoint event needs to be scheduled. If so, the moveThroughControlPoint event is scheduled because the AGV can pass the next control point without reducing its speed. Furthermore, if the moveThroughControlPoint event is executed, the only possible way that event needs to be scheduled is the checkNextPoint event.

When the reachEndOfSegment event is executed, the AGV must go to the end of the segment on which it is traveling. Once entering, it will be added into the waiting list of the next control point and then schedule the travelAlongPath event to trigger itself to start moving.

When the enterControlPoint event is executed, several decisions need to make for scheduling the next appropriate event. There are four situations under which an AGV enters a control point: (1)do loading/unloading operations (2)do battery charging (3)idle and (4)traffic congestion. If the control point which an AGV enters is the destination, then the endofLoading or endOfUnloading event will be scheduled in accordance with the AGV's load status. When an AGV enters a staging area, the chargeAGV event is scheduled for battery charging and no event needs to be scheduled for idle. If an AGV enters a control point due to traffic congestion, the travelAlongPath is scheduled to trigger itself to start moving.

The loading and unloading operations of AGVs are done by the execution of the endOfLoading and endOfUnloading events. The endOfLoading event executes the completion of part loading and then triggers the AGV to find a new path according to the loaded part's routing. Finally, it schedules the travelAlongPath to trigger itself to start moving. However, the endOfUnloading event is somewhat more complicated than the endOfLoading event. When an AGV completes an unloading operation, it first checks its

battery capacity. If the battery is nearly depleted, it will go to a staging area. If not, it then finds another part for transport. It will go to pick it up, if it can find an appropriate one; otherwise, it also goes to a staging area due to idle. All of above situations need to schedule the travelAlongPath event for moving to the desired destination.

So far, all of the events I have discussed are AGV operation events. When AGVs perform these operations, including moving, loading, and unloading, batteries are consumed. Battery capacity is specified by ampere-hours in the system. I also use a number of battery consumption factors to calculate and update AGVs' battery capacity such as traveling, acceleration, deceleration, loading and unloading factors that are specified by ampere-seconds. As soon as the battery capacity of AGVs is thoroughly depleted, breakdown occurs. To accomplish the affects of AGV breakdowns, following steps are implemented:

1. remove the AGV's latest scheduled event from the calendar list because it will cause the AGV to stop immediately
2. temporarily remove the AGV from the AGVSystem so that it will not be requested by parts
3. record the status when the breakdown occurs
4. update its status
5. schedule the maintenance event for repair

The maintenance event is used to resume the AGV's functionalities. To accomplish it, the following steps are required:

1. update its status
2. add this AGV to the AGVSystem
3. schedule the unfinished event which should be performed if breakdown didn't occur
4. schedule the next breakdown

Except for the battery breakdowns, I also schedule general breakdown occurrences in a statistical manner.

AGVSystem:

In the simulation of AGV systems, the AGVSystem object acts as a controller that provides the capabilities to allow other AGV system components to request information. For example, when a part hits the end of output-queue, the ServerStation needs to help this part find an appropriate AGV for transport. In this case, the ServerStation may send a message to AGVSystem to trigger the execution of findAnAppropriateAGV method. This method will then respond according to the AGV selection rule information to the ServerStation. Similarly, when an AGV becomes idle, it may also find a waiting part to transport through the communication with the AGVSystem controller. Additionally, the staging area desiring to find an AGV to trigger it to arrive can accomplish their tasks by doing the same action.

The information of all objects needed to simulate the AGVSystem must be collected in the AGVSystem. Thus, for all AGV system objects to request their desired information, I create global variables to represent the AGVSystem in all AGV system classes.

CHAPTER 4. CASE STUDY AND SIMULATION VERIFICATION/VALIDATION

In this chapter, to verify the Smalltalk/V prototype, I provide two case studies. The first one is to simulate a simple AGV system which is constructed by the traditional layout with the models written in SLAM II and Smalltalk/V. Through the use of hypothesis tests comparing the two different simulation outputs, the validation of the Smalltalk/V computerized model is conducted. In the second case study, I will simulate a more complicated AGV system which is constructed by the tandem layout and then perform a similar analysis.

4.1 Case Study 1

4.1.1 Description of the Target AGV System

The layout of the AGV system used in this case study is shown in Figure 17. Each part arrives to the system at the input station and is transported through the system by either two of AGVs. The processing route of each part is the same and follows the sequence of input station , machine 1, and machine 2. The time between creations of parts is specified by an exponential distribution with a mean of 100 seconds. All of the machine cells have only one server (machine) and a queue size of 999 for their input and output

queues. The processing times of input station, machine cell 1, and machine cell 2 are exponentially distributed with the means of 85, 90, and 95 seconds, respectively. Machine cell 1 has a combined loading/unloading station. The AGVs employed to transport the parts have the same velocity(4.5 ft/s when empty and 4 ft/s when loaded), acceleration(4 ft/s^2, and deceleration (4.5 ft/s^2). In addition, this system has two staging areas where the idle AGVs can wait for the next tasks. The simulation will run from time 0 to time 120000 and the statistics will be collected from time 60000.

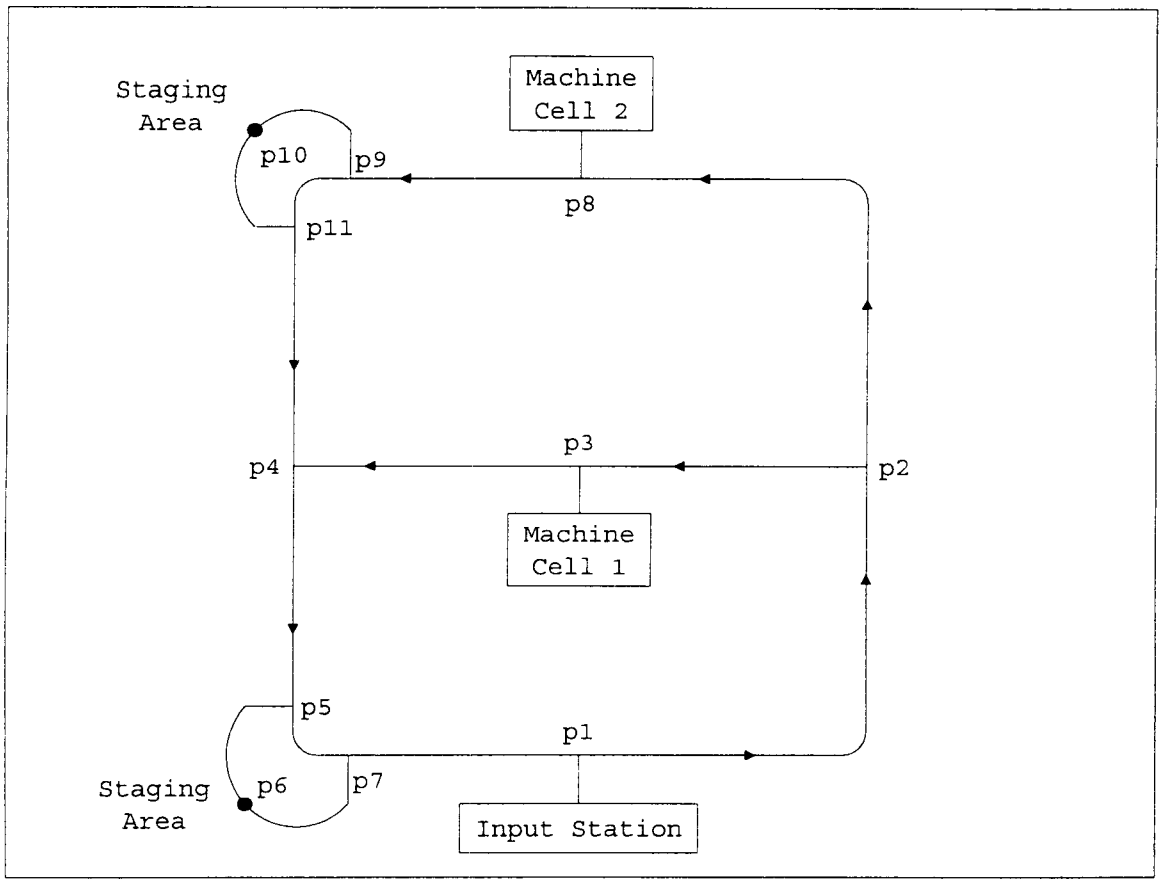


Figure 17: The AGV system of case study 1

4.1.2 SLAM II Simulation Model

To simulate case study 1, I use SLAM II version 4.2 run on a DEC ALPHA UNIX system. The SLAM model and outputs are shown in Appendices A and B.

4.1.3 Smalltalk/V Simulation Model

The modeling of AGV systems is very easy in the object-oriented environment. Beaumariage (1990) presents seven steps to follow in the modeling process. These steps are as follows (Beaumariage 1990):

1. Set up temporary variables which will provide element level (human interaction level) symbols for the element level objects used in the system.
2. Set up a new Calendar class instance.
3. Create instances of classes as needed for the representation of the physical system being modeled and set the temporary variables to point to these instances. Also create instances for terminators and system statistics collection objects.
4. Set up Creator instances for each work flow item type or work order type which will be traveling through the system. This involves specifying (1)the work order and the routings (in terms of the temporary variables mentioned above) through all objects (including statistics collection, material

handling, etc.) which the work flow items will be visiting (in sequential order) and (2) processing times at each location.

5. Set up the list of system elements to include all objects in the system for which output is desired.
6. Schedule any special initial events onto the event list. These might include intermediate results output, clearing of statistics at special times, initial work flow item arrivals, etc.
7. Start the execution of the simulation model by messaging the Calendar object.

The simulation model and outputs are shown in Appendices C and D.

4.1.4 Output Comparisons

Because of minor differences in the operation of modeling constructs, such as AGV movements, the simulation output statistics of SLAM and Smallalk may not be exactly the same. However, through the use of hypothesis tests on those common performance measures of both models, the performance of the two models is comparable. In this case study, since I am interested in all of the performance measures, I perform hypothesis tests on them. The simulation output summary for both models is shown in Tables 1 and 2. I only use the measure of time in system as an

			Input					Mach1					Mach2			AGV		
Run	T. in Sys	To.Obs.	Uti (%)	# in IQ	T. in IQ	# in OQ	T.in OQ	Uti (%)	# in IQ	T. in IQ	# in OQ	T. in OQ	Uti (%)	# in IQ	T.in IQ	Stop (%)	M.E.(%)	M.L.(%)
1	6288.2	584.0	85.6	4.4	427.7	26.1	2512.2	87.3	3.6	375.3	23.5	2435.2	90.7	3.0	309.7	11.60	22.30	66.11
2	3562.7	567.0	83.5	4.1	420.3	7.2	754.7	89.1	4.1	520.7	8.7	927.4	94.8	5.1	538.5	11.70	23.80	64.50
3	8876.4	577.0	92.2	6.4	622.9	59.3	5828.8	91.5	2.6	270.3	16.5	1705.9	90.2	2.7	274.6	11.60	22.30	66.10
4	4681.1	574.0	82.4	3.6	377.9	19.0	1961.3	91.6	7.8	798.9	8.0	829.3	89.0	2.6	267.1	11.65	22.60	65.75
5	4766.2	578.0	74.3	2.1	231.6	20.2	2153.4	87.9	1.7	176.3	13.0	1358.2	90.8	3.7	384.6	11.60	22.50	65.90
6	6494.2	585.0	97.0	19.2	1762.8	32.2	2760.8	88.3	2.6	266.3	9.9	1041.4	90.9	3.1	323.8	11.60	22.45	65.95
7	7677.9	576.0	82.5	3.1	331.1	43.0	4586.6	87.0	2.2	226.6	14.9	1543.8	92.6	4.8	496.8	11.60	23.30	65.10
8	6545.1	592.0	95.9	8.4	811.2	38.6	3674.8	85.2	1.8	187.3	11.4	1168.4	88.7	3.3	354.2	11.60	22.25	66.15
9	2673.0	551.0	79.4	3.0	324.9	4.9	533.2	84.5	3.0	316.3	7.6	840.8	85.9	2.4	254.1	11.95	24.75	63.30
10	6902.5	571.0	81.2	4.4	451.8	42.0	4267.8	89.1	3.6	370.1	10.2	1031.5	91.3	3.7	384.7	11.60	22.50	65.90
11	7388.9	584.0	79.2	3.2	335.8	40.9	4322.3	87.5	2.1	213.1	16.1	1661.7	94.5	3.7	382.9	11.60	22.30	66.10
12	7388.6	579.0	82.4	2.8	291.7	41.3	4228.3	87.8	2.1	220.6	16.0	1660.0	91.8	6.4	659.0	11.60	22.30	66.10
13	5771.6	572.0	80.6	2.4	248.9	26.1	2747.9	92.6	3.3	344.3	15.8	1619.9	88.7	3.3	341.8	11.60	22.30	66.10
14	4604.2	579.0	77.6	2.2	253.8	16.9	1723.4	81.4	1.4	146.2	16.8	1728.8	85.8	2.4	247.5	11.60	22.30	66.10
15	8704.9	570.0	85.5	4.3	449.8	54.1	5677.6	86.9	1.8	190.2	15.3	1582.9	85.6	2.8	283.3	11.60	22.30	66.10
16	7848.5	582.0	95.2	12.5	1186.4	56.0	5305.9	87.9	1.9	197.8	8.2	853.6	89.2	2.0	209.2	11.60	22.30	66.10
17	9222.3	588.0	88.3	4.0	409.0	65.6	6566.4	94.2	3.3	344.7	8.7	906.3	90.6	5.9	612.8	11.60	22.30	66.10
18	4572.2	565.0	78.7	2.0	206.2	18.5	1937.7	85.6	2.1	213.2	10.1	1037.5	93.2	6.7	677.6	11.60	22.30	66.10
19	6930.0	580.0	91.1	4.6	438.3	48.1	4636.4	85.7	2.1	216.3	10.5	1090.8	92.0	3.2	336.1	11.60	22.40	66.00
20	6288.7	576.0	80.1	4.9	510.8	17.6	1814.4	86.0	5.0	517.1	27.2	2818.3	87.3	2.1	218.1	11.60	22.30	66.10
21	7691.3	582.0	81.4	3.4	350.3	39.9	4243.9	82.0	1.1	116.6	23.1	2379.3	83.4	1.7	174.1	11.60	22.30	66.10
22	5458.4	573.0	88.4	8.6	803.4	20.0	2049.8	89.3	5.9	618.3	12.6	1330.7	97.3	4.7	491.4	11.60	22.80	65.60
23	8528.4	561.0	88.2	4.9	481.6	53.3	5266.9	88.5	3.2	323.8	14.4	1484.1	92.0	5.5	569.0	11.65	22.55	65.80
24	4074.1	587.0	80.6	2.4	250.7	15.2	1613.6	90.2	2.3	241.3	10.4	1067.3	91.7	3.6	395.2	11.65	22.45	65.90
25	10251.3	584.0	83.6	5.5	560.4	76.3	7605.0	87.1	2.3	239.1	11.8	1225.6	91.5	2.6	271.1	11.60	22.30	66.10
Avg.	6527.6	576.7	84.6	5.1	501.6	35.3	3550.9	87.8	2.9	306.0	13.6	1413.1	90.4	3.6	378.3	11.62	22.57	65.81
Std D	1905.4	9.2	6.0	3.8	342.4	18.9	1874.1	3.0	1.5	159.6	5.1	523.7	3.2	1.4	143.8	0.07	0.58	0.64

T.in Sys: Time in System
To.Obs : Total Observations
in IQ : Number in Input-Queue
T.in IQ : Time in Input-Queue
in OQ : Number in Output-Queue
T.in OQ : Time in Output-Queue
Uti : Utilization
Stop : Stop State
M.E. : Moving Empty
M.L. : Moving Loaded

Table 1: Smalltalk simulation output summary of case study 1

			Input					Mach1					Mach2			AGV		
Run	T. in Sys	To.Obs.	Uti (%)	# in IQ	T. in IQ	# in OQ	T. in OQ	Uti (%)	# in IQ	T. in IQ	# in OQ	T. in OQ	Uti (%)	# in IQ	T. in IQ	Stop (%)	M.E.(%)	M.L.(%)
1	9890.0	576.0	90.7	7.0	630.8	79.3	6803.1	87.1	2.2	226.2	5.2	541.2	97.0	7.5	774.2	11.60	22.30	66.10
2	5462.0	580.0	82.6	3.5	354.5	23.4	2316.3	82.8	1.4	103.3	16.7	1688.8	91.7	3.2	327.5	11.60	22.30	66.10
3	8295.0	567.0	89.5	4.2	397.1	51.2	4704.8	87.9	2.8	288.0	10.3	1044.1	97.5	11.0	1139.9	11.60	22.30	66.10
4	8301.0	588.0	85.9	3.7	376.1	50.2	4720.9	86.2	2.0	210.6	18.1	1825.4	86.0	2.2	227.9	11.60	22.30	66.10
5	6968.0	577.0	85.6	4.0	332.2	42.1	3970.2	84.4	3.0	309.0	10.9	1116.3	96.9	5.1	528.6	11.60	22.40	66.00
6	9042.0	587.0	91.0	8.9	827.8	59.1	5279.9	83.5	1.1	115.4	15.1	1522.6	89.6	4.9	503.1	11.60	22.30	66.10
7	1936.0	541.0	75.8	3.4	350.6	4.5	553.0	76.2	0.9	101.4	2.5	310.7	82.8	2.1	228.7	12.45	25.65	61.85
8	6924.0	566.0	86.0	8.4	809.3	33.9	3211.4	89.1	2.5	256.0	16.6	1679.9	89.6	4.7	482.7	11.60	22.30	66.10
9	3063.0	564.0	85.6	7.1	732.7	3.3	379.8	87.7	4.3	466.3	6.6	738.3	84.5	1.9	203.3	11.70	24.70	63.60
10	10870.0	579.0	90.6	6.4	625.5	81.6	7279.0	87.4	1.8	191.5	10.3	1047.0	89.6	2.1	218.7	11.60	22.30	66.10
11	8163.0	579.0	84.4	3.8	387.4	43.2	4205.3	92.6	5.5	564.8	13.6	1381.2	93.5	7.3	753.7	11.60	22.30	66.10
12	10500.0	557.0	80.4	3.4	341.3	73.1	6671.2	82.4	1.5	157.7	9.7	986.3	94.4	10.7	1105.6	11.60	22.30	66.10
13	4740.0	589.0	78.3	2.5	261.5	18.9	1920.8	87.0	1.9	197.8	10.2	1068.8	88.8	4.5	455.9	11.60	22.40	65.95
14	7860.0	580.0	77.7	2.6	267.6	47.1	4597.9	88.1	2.7	270.6	13.5	1367.0	88.1	3.3	337.2	11.60	22.30	66.10
15	7410.0	579.0	86.5	3.6	365.9	45.3	4351.1	89.8	3.7	375.4	11.2	1149.1	92.6	3.6	368.3	11.60	22.30	66.10
16	10840.0	582.0	90.9	6.6	635.3	82.3	7350.9	86.3	1.9	198.7	10.2	1045.2	94.0	4.0	409.5	11.60	22.30	66.10
17	2558.0	566.0	78.6	3.2	327.1	7.2	778.8	82.5	1.8	190.7	3.2	359.5	90.4	4.1	436.0	11.90	23.85	64.25
18	4400.0	560.0	81.8	3.9	395.8	14.1	1485.3	90.8	6.6	670.8	11.4	1223.0	84.8	2.1	218.9	12.05	23.80	64.25
19	4759.0	579.0	78.6	1.8	184.6	12.1	1261.3	86.5	2.2	223.4	12.2	1245.9	97.3	11.1	1129.9	11.50	22.75	65.70
20	9688.0	588.0	85.3	4.9	487.6	66.3	6125.8	94.8	5.3	549.3	11.5	1166.3	87.9	2.3	234.9	11.60	22.35	66.00
21	5290.0	570.0	86.9	4.0	383.2	29.6	2836.2	87.7	3.4	351.0	7.8	824.8	91.1	3.2	338.7	11.65	22.50	65.85
22	11280.0	579.0	86.9	8.1	759.1	65.9	5899.1	93.3	4.3	438.9	27.8	2740.5	90.4	6.1	622.7	11.60	22.30	66.10
23	5247.0	573.0	80.4	3.1	325.2	11.8	1276.7	90.6	3.9	396.0	22.7	2294.1	91.0	2.5	262.5	11.50	22.85	65.55
24	5468.0	570.0	94.1	9.6	921.3	28.2	2645.9	87.1	2.3	231.3	7.3	759.8	86.6	3.0	305.5	11.60	22.40	66.00
25	10650.0	575.0	84.8	2.9	273.1	86.9	7638.5	85.8	2.7	275.4	8.1	829.4	89.9	2.9	302.4	11.60	22.35	66.05
Avg.	7184.2	574.0	84.7	4.8	470.1	42.4	3930.5	87.1	2.9	294.4	11.7	1198.2	90.6	4.6	504.0	11.66	22.72	65.61
Std D	2788.5	11.1	4.9	2.2	207.3	26.8	2318.1	3.9	1.4	149.7	5.7	553.8	4.1	2.8	290.6	0.20	0.86	1.04
Sp	2388.1	10.2	5.5	3.1	283.0	23.2	2107.8	3.5	1.5	154.7	5.4	538.9	3.7	2.2	229.2	0.15	0.73	0.86
tp	-1.0	0.9	-0.1	0.3	0.4	-1.1	-0.6	0.7	0.1	0.3	1.3	1.4	-0.3	-1.5	-1.9	-0.90	-0.70	0.79

T.in Sys: Time in System
To.Obs : Total Observations
in IQ : Number in Input-Queue

T.in IQ : Time in Input-Queue
in OQ : Number in Output-Queue
T.in OQ : Time in Output-Queue

Uti : Utilization
Stop : Stop State
M.E. : Moving Empty
M.L. : Moving Loaded

Table 2: SLAM simulation output summary and hypothesis tests of case study 1

example to show how these tests are performed.

Hypothesis test on time in system:

H_0: $\mu_{Smalltalk} = \mu_{SLAM}$

H_1: $\mu_{Smalltalk} \neq \mu_{SLAM}$

$$S_p = \sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}} = 2388.1$$

$$t_p = \frac{\overline{y_1} - \overline{y_2}}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} = -1.0$$

n_1: the simulation runs of Smalltalk

n_2: the simulation runs of SLAM

S_1: standard deviation of time in system of Smalltalk model

S_2: standard deviation of time in system of SLAM model

$\overline{y}_1$: mean value of time in system of Smalltalk model

$\overline{y}_2$: mean value of time in system of SLAM model

S_p: root of a weighted average of the sample variances of the S_1 and S_2 with the degree of freedom of n_1+n_2-2

t_p: value of t distribution for various probability p

Since $|t_p| < t\alpha_{/2}$, do not reject H_0, where $\alpha = 0.05$, $t_{0.025,48} = 2.01$. Refer to Table 2, all of the performance measures' absolute t_p values are less than 2.01. Therefore, based on this result, I conclude that the output from the two models are not significantly different and the models are not different from each other.

4.1.5 Revised Case Study 1

In this section, I modify case study 1 by using small size machine cell queues to compare the simulation results of the original and revised AGV systems. The revised system is only simulated in Smalltalk.

In this revised system, all of the queues of input station, machine cells 1 and 2 have the same size of 5. Except for the queue size, the simulation is run under the same parameters as the original system (refer to section 4.1.1). The outputs of the revised system are shown in Table 3. By comparing Table 1 and Table 3, a summary of the differences between these two systems follows:

- fewer parts completed in the revised system due to the queue size restrictions
- time in system of parts in the revised system is significantly reduced because all queue lengths are shortened
- time in queue and number in queue in the revised system are significantly reduced due to the small size queues
- the AGVs in the revised system have longer idle and moving empty states and have shorter moving loaded states. This is due to system capacity restrictions
- machine cells 1 and 2 of the revised system have lower utilizations

			Input					Mach1					Mach2			AGV		
Run	T. in Sys	To.Obs.	Uti (%)	# in IQ	T. in IQ	# in OQ	T.in OQ	Uti (%)	# in IQ	T. in IQ	# in OQ	T. in OQ	Uti (%)	# in IQ	T.in IQ	Stop (%)	M.E.(%)	M.L.(%)
1	1971.5	521.0	91.7	2.6	293.8	3.8	436.2	93.6	2.2	247.9	3.5	407.1	81.7	1.2	145.7	14.78	25.95	59.27
2	1749.3	491.0	86.0	2.3	277.1	2.9	346.0	84.6	1.6	197.1	2.5	302.3	83.0	1.3	159.9	16.01	27.88	56.11
3	1719.9	519.0	88.3	2.2	257.8	3.3	384.3	83.7	1.5	176.9	2.8	316.8	80.4	1.1	123.1	14.63	26.00	59.37
4	1888.4	507.0	88.8	2.3	278.1	3.4	399.8	91.5	2.1	244.9	3.0	352.8	80.1	1.1	130.5	16.19	25.90	57.91
5	1571.2	519.0	76.0	1.6	183.4	2.7	318.7	87.2	1.5	177.4	2.7	313.5	79.3	1.0	120.2	13.90	27.01	59.09
6	1788.4	528.0	93.4	2.8	316.4	3.2	358.0	83.7	1.6	176.8	3.1	350.9	81.9	1.2	146.6	13.59	26.43	59.98
7	1681.3	517.0	82.4	1.8	217.6	3.0	354.1	84.1	1.5	174.1	2.9	339.8	81.6	1.2	142.4	14.63	26.86	58.51
8	1868.0	519.0	91.6	2.6	299.9	3.2	365.4	89.6	1.9	217.5	3.4	390.9	79.8	1.3	154.0	13.64	27.16	59.20
9	1681.1	510.0	83.0	1.8	215.2	2.7	323.6	86.2	1.6	185.0	3.1	363.0	79.0	1.2	141.4	14.37	27.13	58.50
10	1657.0	509.0	81.0	1.9	224.9	3.0	343.9	84.6	1.4	165.0	2.6	307.7	82.2	1.4	162.5	15.30	26.28	58.42
11	1717.0	507.0	78.7	1.8	217.5	2.7	324.5	87.3	1.7	199.2	2.9	348.0	84.3	1.4	168.2	14.57	27.57	57.86
12	1835.8	516.0	82.7	2.1	250.4	3.2	374.4	87.9	1.9	217.1	3.2	376.3	81.9	1.3	149.9	14.50	26.76	58.74
13	1554.5	497.0	81.7	1.8	213.6	2.8	330.8	86.3	1.5	177.7	2.2	270.0	75.5	0.9	111.9	16.22	26.98	56.80
14	1626.4	511.0	78.9	1.7	196.9	2.8	330.9	84.6	1.6	188.6	2.8	329.6	78.2	1.1	133.6	14.06	27.22	58.72
15	1660.1	516.0	83.2	2.1	242.7	2.7	315.9	86.8	1.5	173.6	3.1	359.6	76.0	1.1	125.9	14.44	27.50	58.06
16	1629.2	530.0	90.2	2.4	365.4	2.5	331.1	85.1	1.3	148.4	2.9	324.5	81.3	1.0	112.5	13.18	26.25	60.57
17	1810.3	520.0	89.5	2.2	258.1	3.3	388.1	87.8	1.7	191.1	3.1	356.1	82.9	1.2	140.9	14.65	25.99	59.36
18	1657.6	502.0	81.6	1.8	211.7	2.9	335.3	84.3	1.8	206.3	2.6	310.1	82.4	1.4	161.9	14.45	27.69	57.86
19	1960.7	534.0	89.5	2.5	291.8	3.3	378.0	92.2	2.1	242.2	3.6	411.3	85.9	1.4	163.6	13.57	26.15	60.28
20	1885.0	516.0	91.7	2.4	281.7	3.4	400.2	89.9	1.9	224.1	3.1	359.8	80.6	1.3	146.9	15.24	25.58	59.18
21	1515.9	527.0	82.9	1.8	199.8	2.7	305.4	83.5	1.5	168.4	2.5	288.9	79.5	0.9	104.6	13.17	26.77	60.06
22	1735.1	513.0	78.9	1.9	228.7	2.8	330.1	88.5	1.6	189.8	3.1	370.4	78.7	1.2	143.3	14.83	27.02	58.15
23	1911.0	516.0	91.0	2.4	279.5	3.4	396.8	89.6	2.1	242.7	3.2	375.8	81.0	1.4	157.0	14.29	27.00	58.71
24	1842.7	529.0	86.6	2.2	250.1	3.2	366.6	91.4	2.0	221.7	3.4	382.5	84.3	1.4	162.8	13.24	26.54	60.22
25	1982.3	497.0	89.4	2.5	294.8	3.1	371.9	90.4	2.0	241.0	3.4	402.6	83.1	1.6	190.0	15.17	27.70	57.14
Avg.	1756.0	514.8	85.5	2.1	253.9	3.0	356.4	87.4	1.7	199.8	3.0	348.4	81.0	1.2	144.0	14.50	26.77	58.72
Std D	134.9	10.8	5.0	0.3	43.8	0.3	32.9	3.0	0.3	29.0	0.3	37.7	2.5	0.2	20.3	0.87	0.64	1.10

T.in Sys: Time in System
To.Obs : Total Observations
in IQ : Number in Input-Queue
T.in IQ : Time in Input-Queue
in OQ : Number in Output-Queue
T.in OQ : Time in Output-Queue
Uti : Utilization
Stop : Stop State
M.E. : Moving Empty
M.L. : Moving Loaded

Table 3: Smalltalk simulation output summary of revised case study 1

4.2 Case Study 2

4.2.1 Description of the Target AGV System

In this section, I simulate a more complicated AGV system by using SLAM II and Smalltalk/V to validate the object-oriented modeling environment. The layout of this AGV system is shown in Figure 18.

This system is a tandem AGV system that has two independent loops and each loop contains only one AGV to transport parts. There are two different types of parts processed through the system: type 1 and type 2. Type 1 has a processing route of input, cell 2, cell 5, cell 7, cell 8, cell 6, and cell 9. The processing times (seconds) at the machine stations are all normal distributions with the following means and standard deviations:(30,6), (100,20), (60,10), (110,15), (110,15), (120,15), and (150,15), respectively. The time between creations of type 1 parts is a normal distribution with a mean of 50 seconds and a standard deviation of 10 seconds. Type 2 parts have a processing route of input, cell 3, cell 4, cell 2, cell 5, cell 6, and cell 9. The processing times (seconds) are all normal distributions with the following means and standard deviations: (60,10), (90,15), (100,15), (100,15), (70,10), (100,15), and (90,15), respectively. The time between creations of type 2 is also a normal distribution with a mean of 200 seconds and a standard deviation of 20 seconds.

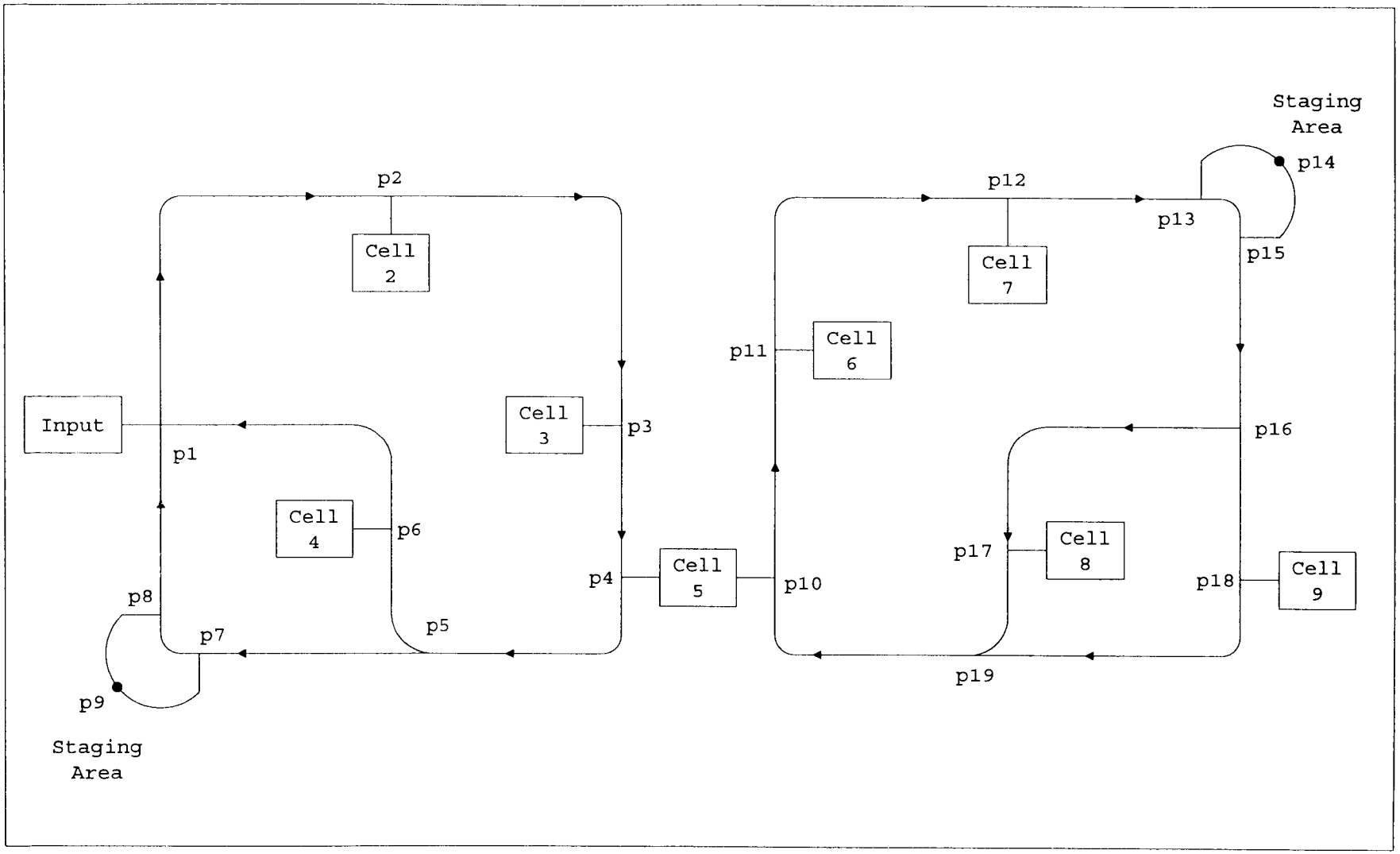


Figure 18: The AGV system layout of case study 2

All of the cells and input station have only one server (machine) and a queue size of 999 for their input and output queues. In addition, I also design one staging area for each loop.

4.2.2 SLAM II Simulation Model

Similarly, I use the same SLAM II version to implement the simulation model for case study 2. The complete SLAM model and outputs are shown in Appendices E and F.

4.2.3 Smalltalk/V Simulation Model

To model case study 2, I use the same modeling procedures discussed previously. The complete model specification and outputs are shown in Appendices G and H.

4.2.4 Output Comparisons

In this case study, since there are too many performance measures, I perform hypothesis tests only on some of them. The simulation output summary for both models is shown in Tables 4 and 5. Similarly, I use the measure of time in system as an example to show how these tests are performed.

												AGV1			AGV2		
Run	T. in Sys	To.Obs.	InputIQL	M2TIQ	M3TOQ	M4%	M5OQL	M6OQL	M7TOQ	M8OQL	M9%	STOP %	M.E. %	M.L. %	STOP %	M.E. %	M.L. %
1	6642.7	22.0	0.5	33.3	0.8	16.0	24.2	1.3	342.7	0.5	78.5	25.1	28.1	46.8	23.9	11.7	64.4
2	6626.5	22.0	0.5	19.3	0.9	17.3	24.4	1.3	346.2	0.5	77.2	25.2	28.0	46.8	23.6	11.6	64.8
3	6586.6	22.0	0.5	10.4	0.8	18.6	23.9	1.4	346.3	0.5	79.0	25.1	27.4	47.6	23.4	11.6	65.0
4	6576.6	21.0	0.7	25.0	0.8	17.4	24.6	1.3	111.7	0.5	75.1	24.9	28.4	46.7	23.7	11.6	64.7
5	6700.1	22.0	0.7	21.0	0.8	17.5	24.4	1.4	332.7	0.5	75.8	25.2	27.5	47.3	23.7	11.1	65.2
6	6556.2	23.0	0.5	10.3	0.8	17.9	24.4	1.4	121.5	0.5	77.3	25.3	27.8	46.9	23.8	11.1	65.2
7	6616.9	22.0	0.5	9.2	0.8	16.8	23.9	1.3	344.3	0.5	78.9	25.2	27.5	47.3	23.7	11.6	64.7
8	6598.5	22.0	0.4	21.1	0.9	17.1	23.8	1.3	359.2	0.5	74.1	25.2	27.5	47.3	23.6	12.0	64.4
9	6534.7	22.0	0.7	10.4	0.9	18.0	24.2	1.4	337.8	0.5	74.8	25.4	27.2	47.4	23.8	11.1	65.2
10	6566.4	22.0	0.6	16.4	0.8	17.7	23.3	1.3	345.6	0.5	78.6	25.3	27.4	47.3	23.7	11.6	64.7
11	6530.5	22.0	0.5	10.2	0.8	17.8	22.5	1.4	118.9	0.5	75.4	25.3	27.0	47.7	23.7	11.6	64.8
12	6496.4	22.0	0.5	7.2	0.9	16.4	24.0	1.4	338.5	0.5	76.8	25.2	27.9	46.9	23.6	11.6	64.8
13	6630.6	22.0	0.4	14.9	0.8	18.4	23.9	1.3	341.9	0.5	77.9	25.1	27.3	47.6	23.6	11.6	64.8
14	6531.1	22.0	0.5	27.5	0.8	16.8	24.2	1.3	348.0	0.5	72.6	25.2	27.4	47.4	23.7	11.6	64.7
15	6600.4	22.0	0.4	21.2	0.8	18.1	24.0	1.3	355.4	0.6	77.0	25.3	27.8	46.9	23.9	11.7	64.4
16	6560.5	22.0	0.4	21.1	0.8	18.2	23.8	1.3	355.2	0.5	75.3	25.2	27.5	47.3	23.9	11.6	64.5
17	6612.3	22.0	0.5	10.8	0.8	17.7	23.6	1.3	350.8	0.5	74.5	25.2	27.5	47.3	23.9	11.7	64.4
18	6590.9	22.0	0.5	25.0	0.8	15.5	23.6	1.4	124.9	0.5	76.3	25.4	27.7	47.0	23.7	11.6	64.7
19	6444.8	22.0	0.4	21.3	0.8	19.1	23.3	1.3	346.8	1.5	76.9	25.3	27.1	47.6	23.7	11.6	64.7
20	6497.8	21.0	0.5	28.0	0.8	17.1	23.6	1.3	110.4	0.5	75.3	25.2	27.7	47.7	23.7	11.6	64.7
21	6637.2	21.0	0.8	13.2	0.9	18.0	23.7	1.3	355.5	0.5	74.2	25.1	27.3	47.6	23.7	11.6	64.7
22	6569.2	22.0	0.5	16.3	0.8	17.3	24.2	1.3	120.9	1.5	74.6	25.2	27.5	47.3	23.8	11.8	64.4
23	6597.4	22.0	0.5	15.2	0.9	15.3	24.2	1.3	351.3	0.5	80.2	25.2	28.4	46.4	23.7	11.6	64.7
24	6695.3	22.0	0.5	15.4	0.8	18.2	25.7	1.3	122.2	0.5	76.6	25.2	27.5	47.3	23.7	11.6	64.8
25	6626.9	22.0	0.5	15.1	0.8	15.1	24.2	1.4	348.8	1.5	75.1	25.2	27.4	47.4	23.7	11.6	64.7
Avg.	6585.1	21.9	0.5	17.5	0.8	17.3	24.0	1.3	283.1	0.6	76.3	25.2	27.6	47.2	23.7	11.6	64.7
Std D	59.5	0.4	0.1	6.8	0.01	1.0	0.6	0.03	104.9	0.3	1.9	0.1	0.4	0.4	0.1	0.2	0.2

T.in Sys: Time in System
To.Obs : Total Observations
InputIQL: Number in Input-Queue of Input Station
TIO: Time in Input-Queue
TOQ: Time in Output-Queue
% : Utilization
OQL: Number in Output-Queue
Stop: Stop State
M.E.: Moving Empty
M.L.: Moving Loaded

Table 4: Smalltalk simulation output summary of case study 2

												AGV1			AGV2		
Run	T. in Sys	To. Obs.	InputIQL	M2TIQ	M3TOQ	M4%	M5OQL	M6OQL	M7TOQ	M8OQL	M9%	STOP %	M.E. %	M.L. %	STOP %	M.E. %	M.L. %
1	6450.0	22.0	0.4	10.3	0.8	18.1	22.7	1.4	321.9	0.6	75.2	25.3	27.4	47.3	23.8	11.6	64.7
2	6587.0	22.0	0.5	30.7	0.8	18.3	24.6	1.3	112.0	0.5	77.5	25.3	27.7	47.0	23.8	11.6	64.7
3	6580.0	22.0	0.4	13.8	0.8	15.5	25.0	1.3	116.3	0.5	74.8	24.9	27.8	47.3	23.7	11.2	65.2
4	6578.0	22.0	0.5	13.5	0.8	19.9	24.4	1.3	124.0	1.5	76.4	25.3	27.4	47.3	23.8	11.6	64.7
5	6628.0	22.0	0.4	16.8	0.8	17.4	24.4	1.4	324.8	0.5	73.7	25.2	27.0	47.8	23.6	11.4	65.0
6	6516.0	22.0	1.1	11.2	0.9	18.2	23.6	1.4	117.7	0.5	78.8	24.9	27.4	47.7	23.8	11.6	64.7
7	6550.0	22.0	0.5	20.0	0.8	16.5	24.5	1.4	319.9	0.5	75.2	25.2	27.4	47.4	23.6	11.4	65.0
8	6636.0	22.0	0.4	18.0	0.9	18.0	23.3	1.4	127.6	1.5	75.5	24.9	27.4	47.7	23.8	11.6	64.7
9	6588.0	22.0	0.6	18.9	0.9	16.9	24.7	1.3	336.3	0.6	76.5	25.3	27.7	47.0	23.8	11.6	64.7
10	6699.0	22.0	0.7	16.0	0.8	19.1	23.8	1.3	317.4	0.5	77.0	25.3	27.4	47.3	23.8	11.6	64.7
11	6596.0	22.0	0.4	12.1	0.9	16.2	23.9	1.3	322.7	0.5	78.5	25.3	27.4	47.3	23.8	11.6	64.7
12	6689.0	21.0	0.7	19.1	0.9	17.9	25.5	1.3	124.9	0.5	78.6	25.0	27.8	47.2	23.7	11.6	64.8
13	6525.0	22.0	0.7	16.7	0.8	17.9	24.7	1.4	112.0	0.5	74.3	24.8	27.6	47.6	23.7	11.6	64.7
14	6639.0	21.0	0.4	39.2	0.8	19.5	24.2	1.4	122.9	1.5	77.0	25.2	27.7	47.1	23.8	11.6	64.7
15	6655.0	22.0	0.5	17.0	0.8	16.4	25.0	1.3	119.6	0.5	78.5	25.3	27.4	47.3	23.8	11.6	64.7
16	6618.0	22.0	0.5	20.7	0.8	17.8	24.2	1.3	336.0	0.5	78.3	25.1	27.4	47.5	23.8	11.6	64.7
17	6515.0	22.0	0.5	13.4	0.8	18.6	25.2	1.3	114.4	0.5	74.7	25.2	27.4	47.4	23.8	11.6	64.7
18	6595.0	22.0	0.4	13.7	0.8	17.2	25.0	1.3	119.9	0.5	76.6	24.9	27.4	47.7	23.6	12.1	64.3
19	6596.0	22.0	0.6	10.8	0.8	17.8	24.4	1.3	315.8	0.5	73.6	25.1	27.8	47.1	23.6	11.5	65.0
20	6662.0	21.0	0.4	21.8	0.8	17.6	25.0	1.4	320.3	0.5	75.2	25.2	27.4	47.4	23.8	11.6	64.7
21	6585.0	22.0	0.5	19.2	0.8	17.0	23.6	1.4	298.2	0.5	73.7	25.0	27.4	47.6	23.8	11.1	65.2
22	6603.0	21.0	0.5	25.8	0.8	18.2	24.0	1.3	320.1	0.5	76.3	25.3	27.7	47.0	23.7	11.6	64.8
23	6599.0	22.0	0.4	12.7	0.8	17.8	22.9	1.3	322.0	0.5	75.9	25.2	27.0	47.8	23.8	12.1	64.2
24	6720.0	23.0	0.5	9.6	0.9	16.5	23.9	1.4	335.1	0.5	75.5	25.3	27.4	47.3	23.8	11.6	64.7
25	6567.0	22.0	0.4	17.6	0.8	16.5	24.9	1.4	299.3	0.5	76.0	25.3	27.4	47.3	23.6	11.1	65.4
Avg.	6599.0	21.9	0.5	17.5	0.8	17.6	24.3	1.3	232.0	0.6	76.1	25.2	27.5	47.4	23.7	11.6	64.8
Std D	61.4	0.4	0.1	6.7	0.01	1.1	0.7	0.03	102.5	0.3	1.6	0.2	0.2	0.2	0.1	0.2	0.3
Sp	60.4	0.4	0.1	6.7	0.01	1.0	0.6	0.03	103.7	0.3	1.7	0.1	0.3	0.3	0.1	0.2	0.2
tp	-0.8	0.3	-0.01	0.002	1.1	-1.0	-1.7	-0.8	1.7	0.01	0.4	1.2	1.5	-1.7	-1.4	-0.1	-0.6

T.in Sys: Time in System
To.Obs : Total Observations
InputIQL: Number in Input-Queue of Input Station
TIO: Time in Input-Queue
TOQ: Time in Output-Queue
% : Utilization
OQL: Number in Output-Queue
Stop: Stop State
M.E.: Moving Empty
M.L.: Moving Loaded

Table 5: SLAM simulation output summary and hypothesis tests of case study 2

Hypothesis test on time in system:

H_0: $\mu_{Smalltalk} = \mu_{SLAM}$

H_1: $\mu_{Smalltalk} \neq \mu_{SLAM}$

$$S_p = \sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}} = 6599.0$$

$$t_p = \frac{\overline{y_1} - \overline{y_2}}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} = -0.8$$

n_1 : the simulation runs of Smalltalk

n_2: the simulation runs of SLAM

S_1: standard deviation of time in system of Smalltalk model

S_2: standard deviation of time in system of SLAM model

$\overline{y_1}$: mean value of time in system of Smalltalk model

$\overline{y_2}$: mean value of time in system of SLAM model

S_p: root of a weighted average of the sample variances of the S_1 and S_2 with the degree of freedom of n_1+n_2-2

t_p: value of t distribution for various probability p

Since $|t_p| < t_{\alpha/2}$, do not reject H_0, where $\alpha = 0.05$, $t_{0.025,48} = 2.01$ Refer to Table 5, since all of the performance measures' absolute t_p values are less than 2.01, therefore, based on this result, I conclude that the output from the two models are not significantly different and the models are not different from each other.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

The object-orientation provides a new and practical approach for simulation modeling and implementations. This research introduces the application of object-oriented simulation modeling to implementations for AGV systems. The information hiding and data abstraction make the simulation of complicated systems easy to develop and extend. I also show that this prototype simulation modeling environment has the ability to simulate both traditional and tandem AGV systems. In addition, compared to conventional simulation tools, object-oriented simulation achieves the trade-off between the flexibility and ease of use.

5.2 Future Work

The following topics should be considered for future work in this area:

1. System functionality enhancements. These include iconic representation, animation capability, bi-directional traffic pattern, and rule option (AGV selection, job dispatching, etc.) extensions.

2. System performance and economic analysis. Due to the flexibility, extendibility, and ease of use, it is ideal to use this system to perform above tasks. For example, the evaluation and analysis of applying various rule options, system layouts, guidepath designs, and direction patterns can be considered. System performance comparisons between traditional and tandem AGV systems can also easily be made. As to economic analysis, we can analyze the manufacturing cost affect resulted from labor allocation due to AGV breakdowns.
3. Integration of this system with other automated material handling systems, such as conveyors, cranes, and AS/RS. By integrating the above systems, a complete material handling capability can be constructed into the original modeling environment. In this research, we simply assume that part movements between the AGV guidepath and machine cells are ignored and the linkage devices between different track loops in tandem AGV systems are machine cells. However, this is not the case in real-world systems. To closely represent these situations, we may use conveyors as the linkage equipment.
4. Application of artificial intelligence (AI) and expert systems. Since the objects contain their own functionality, it has a great potential to build intelligence into this functionality using the machinery of AI and expert systems. For example, an AGV can contain within the definition of its class a decision process for

choosing among various paths to travel. Due to traffic congestion, the shortest path is not guaranteed to be the path with the shortest time to reach the destination. Based on the current situation, an AGV may have the ability to choose an appropriate path through its intelligence. In addition, an AGV may adjust its dispatched task to raise the system performance. Such a situation is shown in Figure 19. At time a, machine cell 1 has part 1 waiting for transport and part 1 has already selected AGV 1. Machine cell 2 has part 2 and part 3 waiting for transport and part 2 has already selected AGV 2. At time b, once AGV 2 loads part 2 and part 3 becomes the first part of the output-queue of machine cell 2, AGV 1 should go to deliver part 3 instead of part 1 because it may raise the system performance, especially if the part routing follows the sequence of machine cell 2 and machine cell 1.

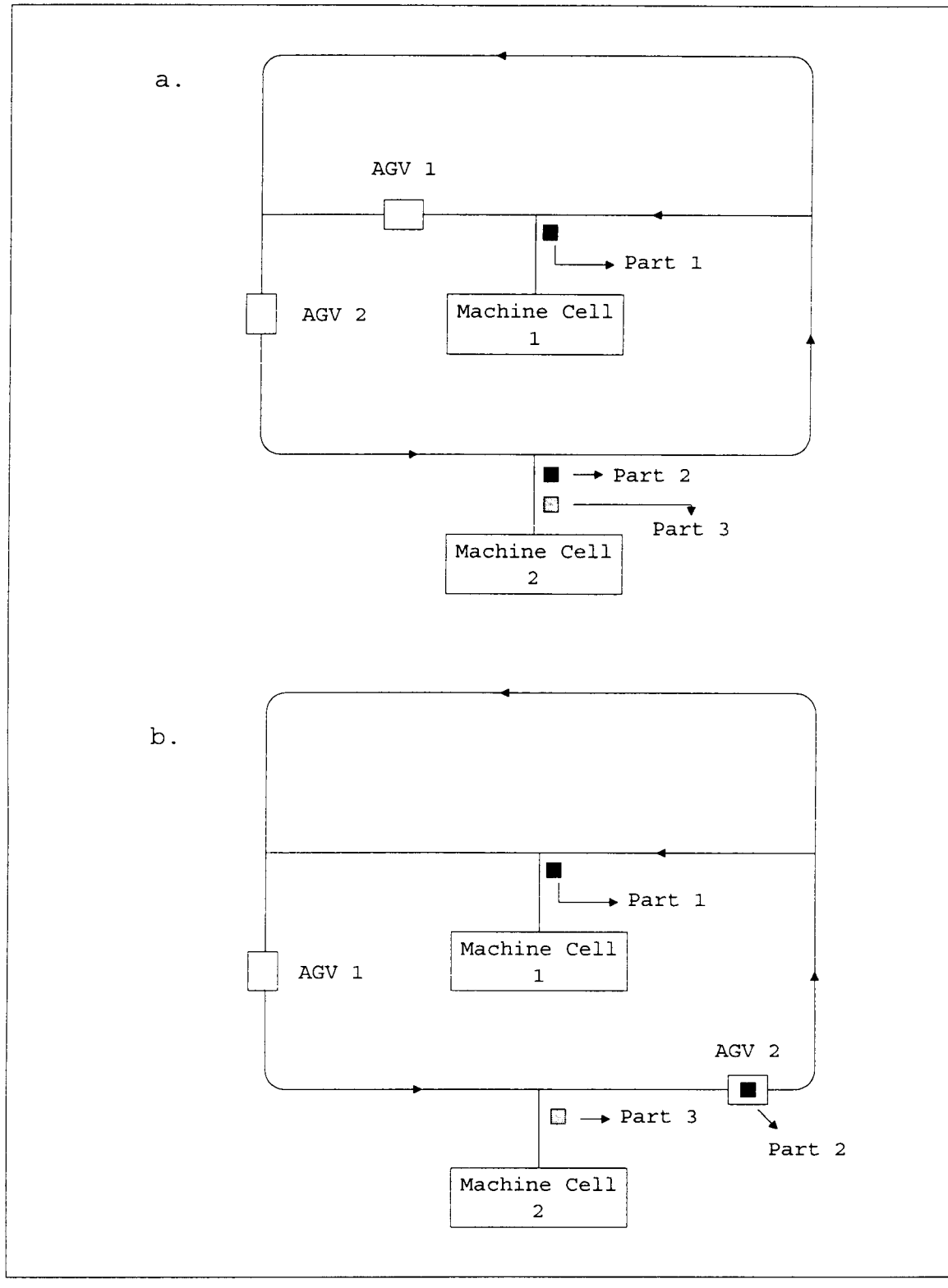


Figure 19: Case of AI and expert system application

BIBLIOGRAPHY

Banks, J., "the Simulation of Material Handling Systems", Simulation, 261-270, Nov., 1990.

Banks, J., "Selecting Simulation Software", Proceedings of 1991 Winter Simulation Conference, 15-20, 1991.

Banks, J., "Software for Simulation", Proceedings of 1993 Winter Simulation Conference, 24-33, 1993.

Bartholdi, John J. and L.K. Platzman, "Decentralized Control of Automated Guided Vehicles on a Simple Loop", IIE Transactions, 21, 1, 76-81, 1989.

Beaumariage, T.G., Investigation of an Object Oriented Modeling Environment for the Generation of Simulation Models, Doctoral Dissertation, Oklahoma State University, 1990.

Bozer, Y.A., and M.M. Srinivasan, "Tandem Configuration for AGV Systems Offers Simplicity and Flexibility", Industrial Engineering, 21, 2, 23-27, 1989.

Bozer, Y.A., and M.M. Srinivasan, "Tandem Configuration for Automated Guided Vehicle System Analysis of Single Vehicle Loops", IIE Transactions, 23, 1, 71-82, 1991.

Choi, H.G., H.J. Kwon, and J. Lee, "Traditional and Tandem AGV System Layouts: A Simulation Study", Simulation, 85-93, august, 1994.

Davis, D.A., "Modeling AGV Systems", Proceedings of the 1986 Winter Simulation Conference, 568-574, 1986.

Dhouib, K. and D. Ait Kadi, "Expert System for AGV Managing in Bidirectional Networks: KADS Methodology Based Approach", International Journal of Production Economics, 33, 31-43, 1994.

Digitalk, Smalltalk/V for Windows Encyclopedia of Classes, Digitalk Inc., 1992.

Digitalk, Smalltalk/V for Windows Tutorial and Programming Handbook, Digitalk Inc., 1992.

Egbelu, P.J., "The Use of Non-Simulation Applications in Estimating Vehicle Requirements in an Automated Guided Vehicle Based Transport System", Material Flow, 4, 17-32, 1987.

Egbelu, P.J. and J.M.A. Tanchoco, "Characterization of Automatic Guided Vehicle Dispatching Rules", International Journal of Production Research, 22, 359-374, 1984.

Egbelu, P.J. and J.M.A. Tanchoco, "Potentials for Bi-Directional Guided Path for Automatic Guided Vehicle Systems", International Journal of Production Research, 24,1075-1099, 1986.

Fishwick, P.A., "Simpack: getting started with simulation programming in C and C++", Proceedings of the 1992 Winter Simulation Conference, 154-162, 1992.

Fujii, S and Sandoh, H., "A Routing Algorithm for Automated Guided Vehicles in FMS", IXth Int. Conf. Prod. Res., Cincinnati, Ohio, 2261-2267, 1987.

Gaskin, R.J. and J.M.A. Tanchoco, "Flow Path Design for Automated Guided Vehicle systems", International Journal of Production Research, 25, 5, 667-676, 1987.

Gaskin, R.J. and J.M.A. Tanchoco, "AGVSim2- a Development Tool for AGVS Controller Design", International Journal of Production Research, 27, 6, 915-926. 1989.

Glavach, M.A. and D.T. Sturrock, "Introduction to SIMAN/CINEMA", Proceedings of the 1993 Winter Simulation Conference, 190-192, 1993.

Goble, J., "Introduction to SIMFACTORY II.5", Proceedings of the 1991 Winter Simulation Conference, 77-80, 1991.

Gong, D.C. and L.F. McGinnis, "an AGVS Simulation Code Generator for Manufacturing Applications", Proceedings of the 1990 Winter Simulation Conference, 676-682, 1990.

Groover, M.P., Automation, Production Systems, and Computer Integrated Manufacturing, Prentice-Hall Internation, Inc., 1987.

Harrel, C.R., "ProModel PC Tutorial", Proceedings of the 1990 Winter Simulation Conference, 128-131, 1990.

Kasales, C.J. and D.T. Strurrock, "Introduction to SIMAN IV", Proceedings of the 1991 Winter Simulation Conference, 106-111, 1991.

Kim Chang, W. and Tanchoco, J.M.A., "Conflict-Free Shortest-Time Bidirectional AGV Routing", International Journal of Production Research, 29, 2377-2391, 1991.

Kulwiec, R.A., Material Handling Handbook, John Wiley & Sons, 1986.

Lee, J., R.H. Choi, and M. Khaksar, "Evaluation of Automated Guided Vehicle Systems by Simulation", Computer and Industrial Engineering, 19, 318-321, 1990.

Lomow, G. and D. Baezner, "A tutorial introduction to object-oriented simulation and Sim++", Proceedings of the 1991 Winter Simulation Conference, 157-163, 1991.

Mahadevan, B. and T.T. Narendran, "Design of as Automated Guided Vehicle-Based Material Handling System for a Flexible Manufacturing System", International Journal of Production Research, 28, 9, 1011-1622, 1990.

Maxwell, W.L. and J.A. Muckstadt, "Design of Automated Guided Vehicle System", IIE Transactions, 14, 2, 114-124, 1982.

McHancy, R., Computer Simulation - a Practical Perspective, Academic Press, Inc., 1991.

Mize, J., H.C. Bhuskute, D.B. Pratt, and M. Kamath, "Modeling of Integrated Manufacturing Systems Using an Object-Oriented Approach", IIE Transactions, 24, 3, 14-25, 1992.

Norman, V.B., "AutoMod II", Proceedings of the 1990 Winter Simulation Conference, 94-98, 1990.

O'Reilly, J.J., "Introduction to SLAM II and SLAMSYSTEM", Proceedings of the 1993 Winter Simulation Conference, 179-183, 1993.

Pegden, C.D. and D.T. Strurrock, "Introduction to SIMAN", Proceedings of the 1990 Winter Simulation Conference, 109-114,1990.

Pristsker, A.A.B., Introduction to Simulation and SLAM II, Third Edition, John Wiley & Sons, New York, 1986.

Production Modeling Corporation, ProModel User Manual, Production Modeling Corporation, 1992.

Rembold, B. and J.M.A. Tanchoco, "a Modular Framework for the Design of Material Flow Systems", International Journal of Production Research, 32, 1, 1-21, 1993.

Roberts, S.D. and J. Heim, "a Perspective on Object Oriented Simulation", Proceedings of the 1988 Winter Simulation Conference, 277-281, 1988.

Roberts, S.D., J.A.Joines, and K.A. Powell, "Object-Oriected Modeling and Simulation with C++", Proceedings of the 1992 Winter Simulation Conference, 145-153, 1992.

Tanchoco, J.M.A., P.J. Egbelu, and F. Taghaboni, "Determination of the Total Number of Vehicles in an AGV-Based Material Transport System", Material Flow, 4, 33-51, 1987.

Tanchoco, J.M.A. and D. Sinriech, "OSL-Optimal Single-Loop Guided Paths for AGVS", International Journal of Production Research, 30, 3, 665-681, 1992.

You, B.A., Modeling and Analyzing Material Handling Systems: an Object Oriented Architecture for Basic Conveyor Simulation, M.S. Thesis, Arizona State University, 1993.

APPENDIX

APPENDIX A: SLAM Simulation Model of Case Study 1

```
GEN,WANG,CASE1,5/24/1995,25;
LIMITS,11,4,1000;
INIT,0.0,120000.0;
VCONT;
;
NETWORK;
      RESOURCE/1,INPUT,1;
      RESOURCE/2,MACH1,5;
      RESOURCE/3,MACH2,8;
;
      VCPOINT,5/P5;
      VCPOINT,9/P9;
      VCPOINT,1/INPUT;
      VCPOINT,6/STAGE1;
      VCPOINT,10/STAGE2;
      VCPOINT,2/P2;
      VCPOINT,7/P7;
      VCPOINT,3/MACH1;
      VCPOINT,11/P11;
      VCPOINT,8/MACH2;
      VCPOINT,4/P4;
;
      VSGMENT,1/S1,1,2,110,,1;
      VSGMENT,7/S7,5,7,20,,1;
      VSGMENT,4/S4,4,5,40,,1;
      VSGMENT,2/S2,2,3,50,,1;
      VSGMENT,5/S5,5,6,13,,1;
      VSGMENT,8/S8,7,1,50,,1;
      VSGMENT,3/S3,3,4,50,,1;
      VSGMENT,6/S6,6,7,13,,1;
      VSGMENT,9/S9,2,8,110,,1;
      VSGMENT,10/S10,8,9,50,,1;
      VSGMENT,13/S13,9,11,13,,1;
      VSGMENT,11/S11,9,10,13,,1;
      VSGMENT,14/S14,11,4,30,,1;
      VSGMENT,12/S12,10,11,13,,1;

VFLEET,AGV,2,4.5,4.0,4.0,4.5,,,,9/CLOSEST,STOP(10,6),6/10;
CR01  CREATE,EXPON(100),,1;
      ACTIVITY;
AW02  AWAIT(1/999),INPUT,BALK(QBALK),1;
```

```
      ACTIVITY/1,EXPON(85);
FR03  FREE,INPUT,1;
      ACTIVITY;
      ASSIGN,XX(5)=NNQ(4);
      ACTIVITY,,XX(5).NE.0;
      ACTIVITY,,XX(5).EQ.0,AS16;
Q18   QUEUE(10),,,;
      ACTIVITY(1)/6,REL(VW04),,AS16;
AS16  ASSIGN,XX(1)=999-NNQ(5),XX(2)=NNRSC(2)+XX(1);
      ACTIVITY,,XX(2).EQ.0;
      ACTIVITY,,XX(2).GT.0,VW04;
Q05   QUEUE(3),,,;
      ACTIVITY(1)/4,REL(FR08),,AS16;
VW04  VWAIT(4),AGV,1,CLOSEST,TOP,1;
      ACTIVITY,6,,VM05;
;
QBALK QUEUE(2),,,;
;
VM05  VMOVE,3;
      ACTIVITY,6;
VF06  VFREE,AGV;
      ACTIVITY;
AW07  AWAIT(5/999),MACH1,,1;
      ACTIVITY/2,EXPON(90);
FR08  FREE,MACH1,1;
      ACTIVITY;
      ASSIGN,XX(6)=NNQ(6);
      ACTIVITY,,XX(6).NE.0;
      ACTIVITY,,XX(6).EQ.0,AS17;
Q19   QUEUE(11),,,;
      ACTIVITY(1)/7,REL(VW10),,AS17;
AS17  ASSIGN,XX(3)=999-NNQ(8),XX(4)=NNRSC(2)+XX(3);
      ACTIVITY,,XX(4).EQ.0;
      ACTIVITY,,XX(4).GT.0,VW10;
Q09   QUEUE(7),,,;
      ACTIVITY(1)/5,REL(FR14),,AS17;
VW10  VWAIT(6),AGV,3,CLOSEST,TOP,1;
      ACTIVITY,6,,VM11;
;
VM11  VMOVE,8;
      ACTIVITY,6;
VF12  VFREE,AGV;
      ACTIVITY;
AW13  AWAIT(8/999),MACH2,,1;
      ACTIVITY/3,EXPON(95);
```

```
FR14   FREE,MACH2,1;
       ACTIVITY;
C15    COLCT,INT(1),TIME IN SYS;
       ENDNETWORK;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
```

```
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
SIMULATE;
MONRE,CLEAR,60000.0;
SIMULATE;
MONTR,CLEAR,60000.0;
FIN;
```

APPENDIX B: SLAM Simulation Outputs of Case Study 1

S L A M I I S U M M A R Y R E P O R T

SIMULATION PROJECT AGV BY WANG

DATE 5/21/1995 RUN NUMBER 1 OF 25

CURRENT TIME 0.1200E+06
STATISTICAL ARRAYS CLEARED AT TIME 0.6000E+05

STATISTICS FOR VARIABLES BASED ON OBSERVATION

	MEAN VALUE	STANDARD DEVIATION	COEFF. OF VARIATION	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBSERVATIONS
TIME IN SYS	0.9890E+04	0.1998E+04	0.2021E+00	0.6543E+04	0.1367E+05	576

FILE STATISTICS

FILE NUMBER	LABEL/TYPE	AVERAGE LENGTH	STANDARD DEVIATION	MAXIMUM LENGTH	CURRENT LENGTH	AVERAGE WAITING TIME
1	AW02 AWAIT	6.9602	5.7048	25	9	630.8330
2	QBAL QUEUE	0.0000	0.0000	0	0	0.0000
3	Q05 QUEUE	0.0000	0.0000	0	0	0.0000
4	VW04 VWAIT	1.0000	0.0000	1	1	103.0928
5	AW07 AWAIT	2.1904	2.3055	9	0	226.2027
6	VW10 VWAIT	0.9875	0.1111	1	1	102.3314
7	Q09 QUEUE	0.0000	0.0000	0	0	0.0000
8	AW13 AWAIT	7.4577	6.3806	26	2	774.1556
9	VEHICLE	1.5412	0.5066	2	1	79.7160
10	Q18 QUEUE	78.2781	20.9178	122	120	6699.9829
11	Q19 QUEUE	4.1990	2.4018	8	7	438.9167
12	CALENDAR	5.7474	0.4864	8	6	9.3252

REGULAR ACTIVITY STATISTICS

ACTIVITY INDEX/LABEL	AVERAGE UTILIZATION	STANDARD DEVIATION	MAXIMUM UTIL	CURRENT UTIL	ENTITY COUNT
1	0.9066	0.2910	1	1	653
2	0.8706	0.3357	1	1	581
3	0.9703	0.1699	1	1	576

SERVICE ACTIVITY STATISTICS

ACTIVITY INDEX	START NODE OR ACTIVITY LABEL	SERVER CAPACITY	AVERAGE UTILIZATION	STANDARD DEVIATION	CURRENT UTILIZATION	AVERAGE BLOCKAGE	MAXIMUM IDLE TIME/SERVERS	MAXIMUM BUSY TIME/SERVERS	ENTITY COUNT
6	Q18 QUEUE	1	1.0000	0.0000	1	0.0000	0.0000	60000.0000	581
4	Q05 QUEUE	1	0.0000	0.0000	0	0.0000	60000.0000	0.0000	0
7	Q19 QUEUE	1	0.9556	0.2059	1	0.0000	549.1875	27491.3672	567
5	Q09 QUEUE	1	0.0000	0.0000	0	0.0000	60000.0000	0.0000	0

RESOURCE STATISTICS

RESOURCE NUMBER	RESOURCE LABEL	CURRENT CAPACITY	AVERAGE UTILIZATION	STANDARD DEVIATION	MAXIMUM UTILIZATION	CURRENT UTILIZATION
1	INPUT	1	0.9066	0.2910	1	1
2	MACH1	1	0.8706	0.3357	1	1
3	MACH2	1	0.9703	0.1699	1	1

RESOURCE NUMBER	RESOURCE LABEL	CURRENT AVAILABLE	AVERAGE AVAILABLE	MINIMUM AVAILABLE	MAXIMUM AVAILABLE
1	INPUT	0	0.0934	0	1
2	MACH1	0	0.1294	0	1
3	MACH2	0	0.0297	0	1

VEHICLE UTILIZATION REPORT

---------- AVERAGE NUMBER OF VEHICLES ----------

VEHICLE FLEET LABEL	NUMBER AVAILABLE	TRAVELING TO LOAD (EMPTY)	LOADING	TRAVELING TO UNLOAD (FULL)	UNLOADING	TOTAL PRODUCTIVE
AGV	2	0.446	0.116	1.322	0.116	2.000

VEHICLE PERFORMANCE REPORT

---------- AVERAGE NUMBER OF VEHICLES ----------

VEHICLE FLEET LABEL	NUMBER OF LOADS	NUMBER OF UNLOADS	TRAVELING EMPTY BLOCKED	TRAVELING FULL BLOCKED	TRAVELING IDLE	STOPPED IDLE	TOTAL NON-PRODUCTIVE
AGV	1159	1159	0.000	0.000	0.000	0.000	0.000

SEGMENT STATISTICS

SEGMENT NUMBER	SEGMENT LABEL	CONTROL END POINTS	NUMBER OF ENTRIES	AVERAGE UTILIZATION	MAXIMUM UTIL.	CURRENT UTIL.
1	S1	1 / 2	1160	0.534	1	0
2	S2	2 / 3	581	0.125	1	0
3	S3	3 / 4	581	0.124	1	0
4	S4	4 / 5	1160	0.182	1	1
5	S5	5 / 6	0	0.000	0	0
6	S6	6 / 7	0	0.000	0	0
7	S7	5 / 7	1159	0.091	1	0

8	S8	7 /	1	1159	0.233	1	0
9	S9	2 /	8	579	0.269	1	1
10	S10	8 /	9	578	0.110	1	0
11	S11	9 /	10	0	0.000	0	0
12	S12	10 /	11	0	0.000	0	0
13	S13	9 /	11	578	0.028	1	0
14	S14	11 /	4	578	0.064	1	0

CONTROL POINT STATISTICS

CONTROL POINT NUMBER	CONTROL POINT LABEL	NUMBER OF ENTRIES	AVERAGE UTILIZATION	MAXIMUM NUMBER WAITING	CURRENT UTILIZATION
1	INPUT	1160	0.068	0	0
2	P2	1160	0.010	0	0
3	MACH1	581	0.121	0	0
4	P4	1160	0.009	0	0
5	P5	1159	0.009	0	0
6	STAGE1	0	0.000	0	0
7	P7	1159	0.009	0	0
8	MACH2	578	0.063	0	0
9	P9	578	0.004	0	0
10	STAGE2	0	0.000	0	0
11	P11	578	0.004	0	0

*** VEHICLE TRIP REPORT MATRIX ***

TABLE 1

FROM CP		TO CP - 1	2	3	4	5	6	7	8	9	10	11	TOTAL
INPUT	1.	0	0	581	0	0	0	0	0	0	0	0	581
P2	2.	0	0	0	0	0	0	0	0	0	0	0	0
MACH1	3.	3	0	578	0	0	0	0	578	0	0	0	1159
P4	4.	0	0	0	0	0	0	0	0	0	0	0	0
P5	5.	0	0	0	0	0	0	0	0	0	0	0	0
STAGE1	6.	0	0	0	0	0	0	0	0	0	0	0	0
P7	7.	0	0	0	0	0	0	0	0	0	0	0	0
MACH2	8.	578	0	0	0	0	0	0	0	0	0	0	578
P9	9.	0	0	0	0	0	0	0	0	0	0	0	0
STAGE2	10.	0	0	0	0	0	0	0	0	0	0	0	0
P11	11.	0	0	0	0	0	0	0	0	0	0	0	0
	TOTAL.	581	0	1159	0	0	0	0	578	0	0	0	2318

Complete outputs are available from Dr. Terrence G. Beaumariage at Oregon State University

APPENDIX C: Smalltalk Simulation Model of Case Study 1

```
|p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 s1 s2 s3
s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 agv1 agv2 input
machine1 machine2 term agvSystem routing1 workOrder1
creater1 calendar i|
i:=0.
25 timesRepeat:[
    i:= i + 1.
    SimOutput
        cr;cr;
        nextPutAll: 'Simulation Output: Run ';
        nextPutAll: i printString;
        nextPutAll: ' of 25';
        cr;cr.
calendar:= Calendar new.
agvSystem:= AGVSystem newWithDispatchingRule: 'Nearest'
withAgvSelectionRule: 'Nearest'.
p0:= ControlPoint newWithName: 'p0'.
p1:= ControlPoint newWithName: 'p1'.
p2:= ControlPoint newWithName: 'p2'.
p3:= ControlPoint newWithName: 'p3'.
p4:= ControlPoint newWithName: 'p4'.
p5:= ControlPoint newWithName: 'p5'.
p6:= StagingArea newWithName: 'p6'.
p7:= ControlPoint newWithName: 'p7'.
p8:= ControlPoint newWithName: 'p8'.
p9:= ControlPoint newWithName: 'p9'.
p10:= StagingArea newWithName: 'p10'.
p11:= ControlPoint newWithName: 'p11'.
p12:= ControlPoint newWithName: 'p12'.

s1:= TrackSegment newWithName: 's1' withStartPoint: p1
     withEndPoint: p2 withLength: 110.
s2:= TrackSegment newWithName: 's2' withStartPoint: p2
     withEndPoint: p3 withLength: 50.
s3:= TrackSegment newWithName: 's3' withStartPoint: p3
     withEndPoint: p4 withLength: 50.
s4:= TrackSegment newWithName: 's4' withStartPoint: p4
     withEndPoint: p5 withLength: 40.
s5:= TrackSegment newWithName: 's5' withStartPoint: p5
     withEndPoint: p6 withLength: 13.
s6:= TrackSegment newWithName: 's6' withStartPoint: p6
```

```
     withEndPoint: p7 withLength:13.
s7:= TrackSegment newWithName: 's7' withStartPoint: p5
     withEndPoint: p7 withLength: 20.
s8:= TrackSegment newWithName: 's8' withStartPoint: p7
     withEndPoint: p1 withLength: 50.
s9:= TrackSegment newWithName: 's9' withStartPoint: p2
     withEndPoint: p8 withLength: 110.
s10:=TrackSegment newWithName:'s10' withStartPoint: p8
     withEndPoint: p9 withLength: 50.
s11:=TrackSegment newWithName:'s11' withStartPoint: p9
     withEndPoint: p10 withLength: 13.
s12:=TrackSegment newWithName:'s12' withStartPoint: p10
     withEndPoint:p11 withLength: 13.
s13:=TrackSegment newWithName:'s13' withStartPoint: p9
     withEndPoint: p11 withLength: 13.
s14:=TrackSegment newWithName:'s14' withStartPoint: p11
     withEndPoint: p4 withLength: 30.

agv1:= AGV newWith: 'AGV1' andCurrentLocation: p6
       andLoadingTime: 6
       andUnloadingTime: 6 andSpeedWhenEmpty: 4.5
       andSpeedWhenLoaded: 4
       andAcceleration: 4 andDeceleration: 4.5
       andBatteryCapacity: 800000
       andTravelEmptyBatteryConsumption:3
       andTravelLoadedBatteryConsumption: 3
       andAccelerationBatteryConsumption: 5
       andDecelerationBatteryConsumption: 6
       andLoadingBatteryConsumption: 5
       andUnloadingBatteryConsumption: 5
       andChargingUnitDuration: 1.2
       andIdleLocation:(Array with: p6 with: p10)
       andTimeBetweenBreakDowns:(Exponential
       newLambda:0.00000001)
       andMaintenanceTime:(Exponential newLambda:0.022).

agv2:= AGV newWith: 'AGV2' andCurrentLocation: p10
       andLoadingTime: 6
       andUnloadingTime: 6 andSpeedWhenEmpty: 4.5
       andSpeedWhenLoaded: 4
       andAcceleration: 4 andDeceleration: 4.5
       andBatteryCapacity: 800000
       andTravelEmptyBatteryConsumption: 3
       andTravelLoadedBatteryConsumption: 3
       andAccelerationBatteryConsumption: 5
```

```
        andDecelerationBatteryConsumption: 6
        andLoadingBatteryConsumption: 5
        andUnloadingBatteryConsumption: 5
        andChargingUnitDuration: 1.2
        andIdleLocation:(Array with: p6 with: p10)
        andTimeBetweenBreakDowns:(Exponential
        newLambda:0.0000000001)
        andMaintenanceTime:(Exponential newLambda:0.022).

input:= InputStation newWithName: 'input' andServerNumber: 1
        andInputQueueSize: 999 andInputLocation:p0
        andOutputQueueSize: 999
        andOutputLocation: p1.
machine1:= SIOQueueMServerProc newWithName: 'm1'
        andServerNumber: 1
        andInputQueueSize:999 andInputLocation: p3
        andOutputQueueSize: 999
        andOutputLocation: p3.
machine2:= SIOQueueMServerProc newWithName: 'm2'
        andServerNumber: 1
        andInputQueueSize:999 andInputLocation: p8
        andOutputQueueSize: 999
        andOutputLocation: p12.
term:= Terminator newWithName:' Final Terminator'.
routing1:= Routing new.
routing1 addOperation: input key: nil
       processingTime:[:rg|rg exponentialLambda:0.011764705]
       setupTime: nil;
       addOperation: machine1 key: nil
       processingTime:[:rg|rg exponentialLambda:0.01111111]
       setupTime: nil;
       addOperation: machine2 key: nil
       processingTime:[:rg|rg exponentialLambda:0.010526315]
       setupTime: nil;
       addOperation: term key: nil.

workOrder1:= WorkOrder newWorkOrderType: 'Work Order 1'.
WorkOrder setWorkOrderNumber: 1.
workOrder1 addComponentWFI:'part 1' andCWFIRouting:routing1.
creater1:= WOCreator newWithWorkOrder: workOrder1
 timeBetweenCreationsGenerator:(Exponential newLambda:0.01).
calendar schedule: [creater1 create] at: 0.
calendar schedule: [calendar clearStatistics] at: 60000.
calendar schedule: [calendar end] at: 120000.
calendar eventInitiator]
```

APPENDIX D: Smalltalk Simulation Output of Case Study 1

Simulation Output: Run 1 of 25

Calendar Statistics

Event List Length Information

Time of initialization = 60000.00

Current Time = 120000

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
8.6355	0.5626	7.0000	5.0000	10.0000	52447

<<< O >>>

Final Terminator Statistics

Time In System Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
584	6288.2194	931.1160	7967.0772	4415.7101	8014.1534

<<< O >>>

Machine Statistics

input (an Input Station Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
621	82.6617	78.1720	116.2011	0.1038	414.4385

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.8555	0.3516	1.0000	0.0000	1.0000	177

Number of Balks Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	0.0000	1

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
4.4398	4.7706	5.0000	0.0000	19.0000	1072

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
621	427.7449	392.3463	399.7295	0.0000	1370.8293

outputQueue information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
26.1447	9.4773	52.0000	9.0000	55.0000	1202

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
580	2512.2253	805.0963	4032.4848	792.2202	4067.9960

<<< O >>>

m1 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
578	90.7196	83.2734	121.8853	0.1278	450.3223

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.8729	0.3331	1.0000	0.0000	1.0000	276

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
3.6172	4.7045	2.0000	0.0000	20.0000	883

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
578	375.3200	485.9925	163.6108	0.0000	2041.2874

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
23.5100	4.8128	25.0000	7.0000	28.0000	1158

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
580	2435.1592	497.8569	2682.4390	792.6431	2897.6987

<<< O >>>

m2 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
584	93.1889	94.5318	39.3725	0.0580	921.8858

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.9071	0.2903	1.0000	0.0000	1.0000	207

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
2.9868	3.3091	1.0000	0.0000	15.0000	958

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
584	309.7338	339.2266	149.6959	0.0000	1475.2014

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	1.0000	1169

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
584	0.0000	0.0000	0.0000	0.0000	0.0000

<<< O >>>

AGV Utilization Statistics

AGV1

States	Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
Stop	0.1160	0.3202	0.0000	0.0000	1.0000	1741
Charging	0.0000	0.0000	0.0000	0.0000	0.0000	1
MovingEmpty	0.2232	0.4164	0.0000	0.0000	1.0000	581
MovingLoaded	0.6608	0.4734	1.0000	0.0000	1.0000	1161

BreakDown Time	Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
	none	none	none	none	none	none

AGV2

States	Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
Stop	0.1159	0.3201	0.0000	0.0000	1.0000	1739
Charging	0.0000	0.0000	0.0000	0.0000	0.0000	1
MovingEmpty	0.2228	0.4162	0.0000	0.0000	1.0000	580
MovingLoaded	0.6613	0.4733	1.0000	0.0000	1.0000	1160

BreakDown Time	Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
	none	none	none	none	none	none

<<< 0 >>>

Segment Utilization Statistics

Name	Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
s1	0.5362	0.4987	0.0000	0.0000	1.0000	2320
s2	0.1251	0.3309	0.0000	0.0000	1.0000	1161
s3	0.1256	0.3314	1.0000	0.0000	1.0000	1160
s4	0.1824	0.3862	0.0000	0.0000	1.0000	2319
s5	0.0000	0.0000	0.0000	0.0000	0.0000	1
s6	0.0000	0.0000	0.0000	0.0000	0.0000	1
s7	0.0912	0.2879	0.0000	0.0000	1.0000	2319
s8	0.2329	0.4227	0.0000	0.0000	1.0000	2319
s9	0.2698	0.4438	1.0000	0.0000	1.0000	1160
s10	0.1127	0.3162	0.0000	0.0000	1.0000	1159

s11	0.0000	0.0000	0.0000	0.0000	0.0000	1
s12	0.0000	0.0000	0.0000	0.0000	0.0000	1
s13	0.0279	0.1646	0.0000	0.0000	1.0000	1159
s14	0.0644	0.2454	0.0000	0.0000	1.0000	1160

<<< 0 >>>

Control Point Utilization Statistics

Name	Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
p0	0.0000	0.0000	0.0000	0.0000	0.0000	1
p1	0.0580	0.2337	0.0000	0.0000	1.0000	2319
p2	0.0000	0.0000	0.0000	0.0000	1.0000	2321
p3	0.1160	0.3202	0.0000	0.0000	1.0000	2321
p4	0.0000	0.0000	0.0000	0.0000	1.0000	2319
p5	0.0000	0.0000	0.0000	0.0000	1.0000	2319
p6	0.0000	0.0000	0.0000	0.0000	0.0000	1
p7	0.0000	0.0000	0.0000	0.0000	1.0000	2319
p8	0.0579	0.2336	0.0000	0.0000	1.0000	1159
p9	0.0000	0.0000	0.0000	0.0000	1.0000	1159
p10	0.0000	0.0000	0.0000	0.0000	0.0000	1
p11	0.0000	0.0000	0.0000	0.0000	1.0000	1159
p12	0.0000	0.0000	0.0000	0.0000	0.0000	1

<<< 0 >>>

Complete outputs are available from Dr. Terrence G. Beaumariage at Oregon State University

APPENDIX E: SLAM Simulation Model of Case Study 2

```
GEN,WANG,CASE2,5/24/1995,25;
LIMITS,27,5,500;
INIT,0.0,10000.0;
VCONT;
;
NETWORK;
      RESOURCE/1,INPUT(1),1;
      RESOURCE/2,MACH2(1),4;
      RESOURCE/3,MACH3(1),5;
      RESOURCE/4,MACH4(1),11;
      RESOURCE/5,MACH5(1),10;
      RESOURCE/6,MACH6(1),17;
      RESOURCE/7,MACH7(1),16;
      RESOURCE/8,MACH8(1),23;
      RESOURCE/9,MACH9(1),20;
;
      VCPOINT,6/P6;
      VCPOINT,18/P18;
      VCPOINT,12/P12;
      VCPOINT,1/P1;
      VCPOINT,7/P7;
      VCPOINT,19/P19;
      VCPOINT,13/P13;
      VCPOINT,2/P2;
      VCPOINT,8/STAG1;
      VCPOINT,14/STAG2;
      VCPOINT,3/P3;
      VCPOINT,9/P9;
      VCPOINT,15/P15;
      VCPOINT,4/P4;
      VCPOINT,10/P10;
      VCPOINT,16/P16;
      VCPOINT,5/P5;
      VCPOINT,11/P11;
      VCPOINT,17/P17;
;
      VSGMENT,1/S1,1,2,60,,1;
      VSGMENT,6/S6,6,1,60,,1;
      VSGMENT,14/S14,12,13,30,,1;
      VSGMENT,10/S10,7,9,15,,1;
      VSGMENT,2/S2,2,3,60,,1;
```

```
       VSGMENT,7/S7,5,7,40,,1;
       VSGMENT,11/S11,9,1,60,,1;
       VSGMENT,15/S15,13,14,15,,1;
       VSGMENT,3/S3,3,4,30,,1;
       VSGMENT,8/S8,7,8,15,,1;
       VSGMENT,4/S4,4,5,35,,1;
       VSGMENT,12/S12,10,11,30,,1;
       VSGMENT,16/S16,14,15,15,,1;
       VSGMENT,9/S9,8,9,15,,1;
       VSGMENT,5/S5,5,6,40,,1;
       VSGMENT,13/S13,11,12,60,,1;
       VSGMENT,17/S17,13,15,15,,1;
       VSGMENT,21/S21,16,17,60,,1;
       VSGMENT,18/S18,15,16,30,,1;
       VSGMENT,19/S19,16,18,50,,1;
       VSGMENT,22/S22,17,19,30,,1;
       VSGMENT,20/S20,18,19,40,,1;
       VSGMENT,23/S23,19,10,50,,1;
;
       VFLEET,AGV,1,4.5,4.0,4.0,4.5,,,,26/CLOSEST,STOP(8),8;
       VFLEET,AGV2,1,4.5,4.0,4.0,4.5,,,,27/CLOSEST,STOP(14),
       14;
;
CR01   CREATE,RNORM(50,10),,2;
       ACTIVITY;
ASS2   ASSIGN,ATRIB(1)=1;
       ACTIVITY;
AW03   AWAIT(1/999),INPUT;
       ACTIVITY/1,RNORM(30,6),ATRIB(1).EQ.1;
       ACTIVITY/2,RNORM(60,10),ATRIB(1).EQ.2;
FR03   FREE,INPUT;
       ACTIVITY;
Q09    QUEUE(2),,,;
       ACTIVITY(1);
VW10   VWAIT(3),AGV,1,CLOSEST;
       ACTIVITY,5.9,ATRIB(1).EQ.1;
       ACTIVITY,5.9,ATRIB(1).EQ.2,VM12;
VM11   VMOVE,2;
       ACTIVITY,5.9;
VF13   VFREE,AGV;
       ACTIVITY;
AW15   AWAIT(4/999),MACH2;
       ACTIVITY/3,RNORM(100,20),ATRIB(1).EQ.1,FR17;
       ACTIVITY/4,RNORM(100,15),ATRIB(1).EQ.2,FR17;
VM12   VMOVE,3;
```

```
      ACTIVITY,5.9;
VF14  VFREE,AGV;
      ACTIVITY;
AW16  AWAIT(5/999),MACH3;
      ACTIVITY/5,RNORM(90,15),,FR18;
;
CR05  CREATE,RNORM(200,20),,2;
      ACTIVITY;
ASS6  ASSIGN,ATRIB(1)=2;
      ACTIVITY,,,AW03;
;
FR17  FREE,MACH2;
      ACTIVITY;
Q19   QUEUE(6),,999,;
      ACTIVITY(1);
VW21  VWAIT(8),AGV,2,CLOSEST;
      ACTIVITY,5.9;
VM23  VMOVE,4;
      ACTIVITY,6;
VF25  VFREE,AGV;
      ACTIVITY;
AW27  AWAIT(10/999),MACH5;
      ACTIVITY/6,RNORM(60,10),ATRIB(1).EQ.1;
      ACTIVITY/7,RNORM(70,10),ATRIB(1).EQ.2;
FR29  FREE,MACH5;
      ACTIVITY;
Q31   QUEUE(14),,,;
      ACTIVITY(1),,,VW33;
;
FR18  FREE,MACH3;
      ACTIVITY;
Q20   QUEUE(7),,,;
      ACTIVITY(1);
VW22  VWAIT(9),AGV,3,CLOSEST;
      ACTIVITY,6;
VM24  VMOVE,6;
      ACTIVITY,6;
VF26  VFREE,AGV;
      ACTIVITY;
AW28  AWAIT(11/999),MACH4;
      ACTIVITY/8,RNORM(100,15);
FR30  FREE,MACH4;
      ACTIVITY;
Q30   QUEUE(12),,,;
      ACTIVITY(1);
```

```
VW32  VWAIT(13),AGV,6,CLOSEST;
      ACTIVITY,6,,VM34;
;
VM34  VMOVE,2;
      ACTIVITY,6;
VF59  VFREE,AGV;
      ACTIVITY,,,AW15;
;
VW33  VWAIT(15),AGV2,10,CLOSEST;
      ACTIVITY,6,ATRIB(1).EQ.1;
      ACTIVITY,6,ATRIB(1).EQ.2,VM36;
VM35  VMOVE,12;
      ACTIVITY,6;
VF37  VFREE,AGV2;
      ACTIVITY;
AW39  AWAIT(16/999),MACH7;
      ACTIVITY/9,RNORM(110,15);
FR41  FREE,MACH7;
      ACTIVITY;
Q50   QUEUE(21),,,;
      ACTIVITY,,,VW51;
VM36  VMOVE,11;
      ACTIVITY,6;
VF38  VFREE,AGV2;
      ACTIVITY;
AW40  AWAIT(17/999),MACH6;
      ACTIVITY/10,RNORM(120,15),ATRIB(1).EQ.1;
      ACTIVITY/11,RNORM(100,15),ATRIB(1).EQ.2;
FR42  FREE,MACH6;
      ACTIVITY;
Q43   QUEUE(18),,999,;
      ACTIVITY(1),,,VW44;
;
VW44  VWAIT(19),AGV2,11,CLOSEST;
      ACTIVITY,6;
VM45  VMOVE,18;
      ACTIVITY,6;
VF53  VFREE,AGV2;
      ACTIVITY;
AW46  AWAIT(20/999),MACH9;
      ACTIVITY/12,RNORM(150,15),ATRIB(1).EQ.1;
      ACTIVITY/13,RNORM(90,15),ATRIB(1).EQ.2;
FR48  FREE,MACH9;
      ACTIVITY;
C49   COLCT,INT(2),TIME IN SYSTEM;
```

```
      TERMINATE;
;
VW51  VWAIT(22),AGV2,12,CLOSEST;
      ACTIVITY,6;
VM52  VMOVE,17;
      ACTIVITY,6;
VF54  VFREE,AGV2;
      ACTIVITY;
AW55  AWAIT(23/999),MACH8;
      ACTIVITY/14,RNORM(110,15);
FR56  FREE,MACH8;
      ACTIVITY;
Q57   QUEUE(24),,999,;
      ACTIVITY(1);
VW58  VWAIT(25),AGV2,17,CLOSEST;
      ACTIVITY,6;
VM60  VMOVE,11;
      ACTIVITY,6;
VF61  VFREE,AGV2;
      ACTIVITY,,,AW40;
      ENDWORKS;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMUALTE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
```

```
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMUALTE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
SIMULATE;
MONTR,CLEAR,6000.0;
FIN;
```

APPENDIX F: SLAM Simulation Outputs of Case Study 2

S L A M I I S U M M A R Y R E P O R T

SIMULATION PROJECT AGV BY WANG

DATE 5/21/1995 RUN NUMBER 1 OF 25

CURRENT TIME 0.1000E+05
STATISTICAL ARRAYS CLEARED AT TIME 0.6000E+04

STATISTICS FOR VARIABLES BASED ON OBSERVATION

	MEAN VALUE	STANDARD DEVIATION	COEFF. OF VARIATION	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBSERVATIONS
TIME IN SYSTEM	0.6607E+04	0.9059E+03	0.1371E+00	0.5129E+04	0.8054E+04	22

FILE STATISTICS

FILE NUMBER	LABEL/TYPE	AVERAGE LENGTH	STANDARD DEVIATION	MAXIMUM LENGTH	CURRENT LENGTH	AVERAGE WAITING TIME
1	AW03 AWAIT	0.5392	0.5765	2	1	20.9412
2	Q09 QUEUE	0.0000	0.0000	0	0	0.0000
3	VW10 VWAIT	130.5131	19.2066	163	163	2623.3794
4	AW15 AWAIT	0.2087	0.4064	1	1	22.5651
5	AW16 AWAIT	0.0000	0.0000	1	0	0.0000
6	Q19 QUEUE	0.0000	0.0000	0	0	0.0000
7	Q20 QUEUE	0.0000	0.0000	0	0	0.0000
8	VW21 VWAIT	0.8833	0.5359	2	0	95.4919
9	VW22 VWAIT	0.8456	0.3613	1	1	422.8210
10	AW27 AWAIT	0.0000	0.0000	1	0	0.0000
11	AW28 AWAIT	0.0000	0.0000	1	0	0.0000
12	Q30 QUEUE	0.0000	0.0000	0	0	0.0000

13	VW32	VWAIT	0.1327	0.3392	1	0	75.8154
14	Q31	QUEUE	0.0000	0.0000	0	0	0.0000
15	VW33	VWAIT	24.1367	4.0265	31	31	1821.6368
16	AW39	AWAIT	0.0000	0.0000	1	0	0.0000
17	AW40	AWAIT	0.0211	0.1438	1	0	3.6716
18	Q43	QUEUE	0.0000	0.0000	0	0	0.0000
19	VW44	VWAIT	1.3348	0.5147	2	1	222.4688
20	AW46	AWAIT	0.1037	0.3048	1	0	18.8489
21	Q50	QUEUE	0.0000	0.0000	0	0	0.0000
22	VW51	VWAIT	1.5308	0.4990	2	2	306.1642
23	AW55	AWAIT	0.0000	0.0000	1	0	0.0000
24	Q57	QUEUE	0.0000	0.0000	0	0	0.0000
25	VW58	VWAIT	1.4953	0.5000	2	1	314.8070
26		VEHICLE	132.1048	19.0211	164	164	2105.2556
27		VEHICLE	28.3877	4.0221	36	35	978.8846
28		CALENDAR	9.1300	0.9233	12	10	9.4148

REGULAR ACTIVITY STATISTICS

ACTIVITY INDEX/LABEL	AVERAGE UTILIZATION	STANDARD DEVIATION	MAXIMUM UTIL	CURRENT UTIL	ENTITY COUNT
1	0.5861	0.4925	1	0	82
2	0.3244	0.4681	1	1	20
3	0.7246	0.4467	1	1	28
4	0.1833	0.3869	1	0	7
5	0.1544	0.3613	1	0	8
6	0.4283	0.4948	1	0	29
7	0.1299	0.3362	1	0	7
8	0.1745	0.3796	1	1	7
9	0.4692	0.4990	1	0	18
10	0.5202	0.4996	1	1	18
11	0.1239	0.3294	1	0	5
12	0.7096	0.4539	1	1	18
13	0.0970	0.2960	1	0	4
14	0.5047	0.5000	1	1	17

SERVICE ACTIVITY STATISTICS

ACTIVITY INDEX	START NODE OR ACTIVITY LABEL	SERVER CAPACITY	AVERAGE UTILIZATION	STANDARD DEVIATION	CURRENT UTILIZATION	AVERAGE BLOCKAGE	MAXIMUM IDLE TIME/SERVERS	MAXIMUM BUSY TIME/SERVERS	ENTITY COUNT
0	Q09 QUEUE	1	0.0000	0.0000	0	0.0000	85.3857	0.0000	
0	Q19 QUEUE	1	0.0000	0.0000	0	0.0000	202.2573	0.0000	
0	Q31 QUEUE	1	0.0000	0.0000	0	0.0000	202.3906	0.0000	
0	Q20 QUEUE	1	0.0000	0.0000	0	0.0000	622.5781	0.0000	
0	Q30 QUEUE	1	0.0000	0.0000	0	0.0000	638.6143	0.0000	
0	Q50 QUEUE	1	0.0000	0.0000	0	0.0000	327.8208	0.0000	
0	Q43 QUEUE	1	0.0000	0.0000	0	0.0000	242.1294	0.0000	
0	Q57 QUEUE	1	0.0000	0.0000	0	0.0000	317.7295	0.0000	

RESOURCE STATISTICS

RESOURCE NUMBER	RESOURCE LABEL	CURRENT CAPACITY	AVERAGE UTILIZATION	STANDARD DEVIATION	MAXIMUM UTILIZATION	CURRENT UTILIZATION
1	INPUT	1	0.9104	0.2856	1	1
2	MACH2	1	0.9080	0.2891	1	1
3	MACH3	1	0.1544	0.3613	1	0
4	MACH4	1	0.1745	0.3796	1	1
5	MACH5	1	0.5582	0.4966	1	0
6	MACH6	1	0.6441	0.4788	1	1
7	MACH7	1	0.4692	0.4990	1	0
8	MACH8	1	0.5047	0.5000	1	1
9	MACH9	1	0.8067	0.3949	1	1

RESOURCE NUMBER	RESOURCE LABEL	CURRENT AVAILABLE	AVERAGE AVAILABLE	MINIMUM AVAILABLE	MAXIMUM AVAILABLE
1	INPUT	0	0.0896	0	1
2	MACH2	0	0.0920	0	1
3	MACH3	1	0.8456	0	1
4	MACH4	0	0.8255	0	1
5	MACH5	1	0.4418	0	1
6	MACH6	0	0.3559	0	1
7	MACH7	1	0.5308	0	1
8	MACH8	0	0.4953	0	1
9	MACH9	0	0.1933	0	1

VEHICLE UTILIZATION REPORT

---------- AVERAGE NUMBER OF VEHICLES ----------

VEHICLE FLEET LABEL	NUMBER AVAILABLE	TRAVELING TO LOAD (EMPTY)	LOADING	TRAVELING TO UNLOAD (FULL)	UNLOADING	TOTAL PRODUCTIVE
AGV	1	0.270	0.131	0.469	0.131	1.000
AGV2	1	0.110	0.122	0.647	0.122	1.000

VEHICLE PERFORMANCE REPORT

---------- AVERAGE NUMBER OF VEHICLES ----------

VEHICLE FLEET LABEL	NUMBER OF LOADS	NUMBER OF UNLOADS	TRAVELING EMPTY BLOCKED	TRAVELING FULL BLOCKED	TRAVELING IDLE	STOPPED IDLE	TOTAL NON-PRODUCTIVE
AGV	87	87	0.000	0.000	0.000	0.000	0.000
AGV2	81	81	0.000	0.000	0.000	0.000	0.000

SEGMENT STATISTICS

SEGMENT NUMBER	SEGMENT LABEL	CONTROL END POINTS	NUMBER OF ENTRIES	AVERAGE UTILIZATION	MAXIMUM UTIL.	CURRENT UTIL.
1	S1	1 / 2	44	0.156	1	0
2	S2	2 / 3	44	0.155	1	1
3	S3	3 / 4	43	0.079	1	0
4	S4	4 / 5	43	0.077	1	0
5	S5	5 / 6	43	0.097	1	0

6	S6	6 / 1	43	0.143	1	0
7	S7	5 / 7	0	0.000	0	0
8	S8	7 / 8	0	0.000	0	0
9	S9	8 / 9	0	0.000	0	0
10	S10	7 / 9	0	0.000	0	0
11	S11	9 / 1	0	0.000	0	0
12	S12	10 / 11	40	0.071	1	0
13	S13	11 / 12	41	0.148	1	0
14	S14	12 / 13	41	0.072	1	0
15	S15	13 / 14	0	0.000	0	0
16	S16	14 / 15	0	0.000	0	0
17	S17	13 / 15	41	0.038	1	0
18	S18	15 / 16	41	0.075	1	1
19	S19	16 / 18	22	0.069	1	0
20	S20	18 / 19	22	0.044	1	0
21	S21	16 / 17	18	0.068	1	0
22	S22	17 / 19	18	0.029	1	0
23	S23	19 / 10	40	0.117	1	0

CONTROL POINT STATISTICS

CONTROL POINT NUMBER	CONTROL POINT LABEL	NUMBER OF ENTRIES	AVERAGE UTILIZATION	MAXIMUM NUMBER WAITING	CURRENT UTILIZATION
1	P1	43	0.075	0	0

2	P2	44	0.133	0	0
3	P3	43	0.043	0	0
4	P4	43	0.074	0	0
5	P5	43	0.019	0	0
6	P6	43	0.040	0	0
7	P7	0	0.000	0	0
8	STAG1	0	0.000	0	0
9	P9	0	0.000	0	0
10	P10	40	0.053	0	0
11	P11	41	0.090	0	0
12	P12	41	0.075	0	0
13	P13	41	0.021	0	0
14	STAG2	0	0.000	0	0
15	P15	41	0.020	0	1
16	P16	40	0.020	0	0
17	P17	18	0.063	0	0
18	P18	22	0.043	0	0
19	P19	40	0.019	0	0

*** VEHICLE TRIP REPORT MATRIX ***

FROM CP \ TO CP -	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	TOTAL
P1 1.	0	30	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37

P2	2.	0	37	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73
P3	3.	0	0	7	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	14
P4	4.	29	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	36
P5	5.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P6	6.	7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14
P7	7.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAG1	8.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P9	9.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P10	10.	0	0	0	0	0	0	0	0	0	0	5	18	0	0	0	0	0	0	0	23
P11	11.	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	22	0	45
P12	12.	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	18	0	0	36
P13	13.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAG2	14.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P15	15.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P16	16.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P17	17.	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	18	0	0	36
P18	18.	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	22
P19	19.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TOTAL.	36	74	14	36	0	14	0	0	0	22	46	36	0	0	0	0	36	22	0	336

Complete outputs are available from Dr. Terrence G. Beaumariage at Oregon State University

APPENDIX G: Smalltalk Simulation Model of Case Study 2

```
|p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 p15 p16
p17 p18 p19 p20 s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13
s14 s15 s16 s17 s18 s19 s20 s21 s22 s23 agv1 agv2 input
machine1 machine2 machine3 machine4 machine5 machine6
machine7 machine8 machine9 machine10 term agvSystem routing1
routing2 workOrder1 workOrder2 creater1 creater2 calendar i|
i:=0.
25 timesRepeat:[
    i:= i + 1.
    SimOutput
        cr;cr;
        nextPutAll: 'Simulation Output: Run ';
        nextPutAll: i printString;
        nextPutAll: ' of 25';
        cr;cr.
calendar:= Calendar new.
agvSystem:= AGVSystem newWithDispatchingRule: 'Nearest'
                      withAgvSelectionRule: 'Nearest'.
p0:= ControlPoint newWithName: 'p0'.
p1:= ControlPoint newWithName: 'p1'.
p2:= ControlPoint newWithName: 'p2'.
p3:= ControlPoint newWithName: 'p3'.
p4:= ControlPoint newWithName: 'p4'.
p5:= ControlPoint newWithName: 'p5'.
p6:= ControlPoint newWithName: 'p6'.
p7:= ControlPoint newWithName: 'p7'.
p8:= StagingArea newWithName: 'p8'.
p9:= ControlPoint newWithName: 'p9'.
p10:= ControlPoint newWithName: 'p10'.
p11:= ControlPoint newWithName: 'p11'.
p12:= ControlPoint newWithName: 'p12'.
p13:= ControlPoint newWithName: 'p13'.
p14:= StagingArea newWithName: 'p14'.
p15:= ControlPoint newWithName: 'p15'.
p16:= ControlPoint newWithName: 'p16'.
p17:= ControlPoint newWithName: 'p17'.
p18:= ControlPoint newWithName: 'p18'.
p19:= ControlPoint newWithName: 'p19'.
p20:= ControlPoint newWithName: 'p20'.
s1:= TrackSegment newWithName: 's1' withStartPoint: p1
     withEndPoint: p2 withLength: 60.
```

```
s2:= TrackSegment newWithName: 's2' withStartPoint: p2
     withEndPoint: p3 withLength: 60.
s3:= TrackSegment newWithName: 's3' withStartPoint: p3
     withEndPoint: p4 withLength: 30.
s4:= TrackSegment newWithName: 's4' withStartPoint: p4
     withEndPoint: p5 withLength: 35.
s5:= TrackSegment newWithName: 's5' withStartPoint: p5
     withEndPoint: p6 withLength: 40.
s6:= TrackSegment newWithName: 's6' withStartPoint: p6
     withEndPoint: p1 withLength:60.
s7:= TrackSegment newWithName: 's7' withStartPoint: p5
     withEndPoint: p7  withLength:40.
s8:= TrackSegment newWithName: 's8' withStartPoint: p7
     withEndPoint: p8 withLength: 15.
s9:= TrackSegment newWithName: 's9' withStartPoint: p8
     withEndPoint: p9 withLength: 15.
s10:= TrackSegment newWithName: 's10' withStartPoint: p7
     withEndPoint:p9 withLength: 15.
s11:= TrackSegment newWithName: 's11' withStartPoint: p9
     withEndPoint:p1 withLength: 60.
s12:= TrackSegment newWithName:'s12' withStartPoint:p10
     withEndPoint:p11 withLength: 30.
s13:= TrackSegment newWithName:'s13' withStartPoint:p11
     withEndPoint:p12 withLength: 60.
s14:= TrackSegment newWithName:'s14' withStartPoint:p12
     withEndPoint:p13 withLength: 30.
s15:= TrackSegment newWithName:'s15' withStartPoint:p13
     withEndPoint:p14 withLength: 15.
s16:= TrackSegment newWithName:'s16' withStartPoint:p14
     withEndPoint:p15 withLength: 15.
s17:= TrackSegment newWithName:'s17' withStartPoint:p13
     withEndPoint:p15 withLength: 15.
s18:= TrackSegment newWithName:'s18' withStartPoint:p15
     withEndPoint:p16 withLength: 30.
s19:= TrackSegment newWithName:'s19' withStartPoint:p16
     withEndPoint:p18 withLength: 50.
s20:= TrackSegment newWithName:'s20' withStartPoint:p18
     withEndPoint:p19 withLength: 40.
s21:= TrackSegment newWithName:'s21' withStartPoint:p16
     withEndPoint:p17 withLength: 60.
s22:= TrackSegment newWithName:'s22' withStartPoint:p17
     withEndPoint:p19 withLength: 30.
s23:= TrackSegment newWithName:'s23' withStartPoint:p19
     withEndPoint:p10 withLength: 50.
```

```
agv1:=AGV newWith: 'AGV1' andCurrentLocation: p8
     andLoadingTime: 6 andUnloadingTime: 6
     andSpeedWhenEmpty: 4.5 andSpeedWhenLoaded: 4
     andAcceleration: 4 andDeceleration: 4.5
     andBatteryCapacity: 80000
     andTravelEmptyBatteryConsumption: 3
     andTravelLoadedBatteryConsumption: 3
     andAccelerationBatteryConsumption: 5
     andDecelerationBatteryConsumption: 6
     andLoadingBatteryConsumption: 5
     andUnloadingBatteryConsumption: 5
     andChargingUnitDuration: 2
     andIdleLocation: (Array with: p8)
     andTimeBetweenBreakDowns:(Exponential
     newLambda:0.00000022)
     andMaintenanceTime:(Exponential newLambda:0.22).

agv2:=AGV newWith: 'AGV2' andCurrentLocation: p14
     andLoadingTime: 6 andUnloadingTime: 6
     andSpeedWhenEmpty: 4.5 andSpeedWhenLoaded: 4
     andAcceleration: 4 andDeceleration: 4.5
     andBatteryCapacity: 80000
     andTravelEmptyBatteryConsumption: 3
     andTravelLoadedBatteryConsumption: 3
     andAccelerationBatteryConsumption: 5
     andDecelerationBatteryConsumption: 6
     andLoadingBatteryConsumption: 5
     andUnloadingBatteryConsumption: 5
     andChargingUnitDuration: 2
     andIdleLocation: (Array with: p14 )
     andTimeBetweenBreakDowns:(Exponential
     newLambda:0.00000022)
     andMaintenanceTime:(Exponential newLambda:0.22 ).

input:= InputStation newWithName: 'input' andServerNumber: 1
     andInputQueueSize: 999 andInputLocation: p0
     andOutputQueueSize:999 andOutputLocation: p1.
machine2:=SIOQueueMServerProc newWithName:'m2'
     andServerNumber: 1
     andInputQueueSize: 999 andInputLocation: p2
     andOutputQueueSize:999 andOutputLocation: p2.
machine3:=SIOQueueMServerProc newWithName: 'm3'
     andServerNumber: 1
     andInputQueueSize: 999 andInputLocation: p3
     andOutputQueueSize:999 andOutputLocation: p3.
```

```
machine4:=SIOQueueMServerProc newWithName: 'm4'
     andServerNumber: 1 andInputQueueSize: 999
     andInputLocation: p6 andOutputQueueSize:999
     andOutputLocation: p6.
machine5:=SIOQueueMServerProc newWithName: 'm5'
     andServerNumber: 1 andInputQueueSize: 999
     andInputLocation: p4 andOutputQueueSize:999
     andOutputLocation: p10.
machine6:=SIOQueueMServerProc newWithName: 'm6'
     andServerNumber: 1 andInputQueueSize: 999
     andInputLocation:p11 andOutputQueueSize:999
     andOutputLocation: p11.
machine7:=SIOQueueMServerProc newWithName: 'm7'
     andServerNumber: 1 andInputQueueSize: 999
     andInputLocation:p12 andOutputQueueSize:999
     andOutputLocation: p12.
machine8:=SIOQueueMServerProc newWithName: 'm8'
     andServerNumber: 1 andInputQueueSize: 999
     andInputLocation:p17 andOutputQueueSize:999
     andOutputLocation: p17.
machine9:=SIOQueueMServerProc newWithName: 'm9'
     andServerNumber: 1 andInputQueueSize: 999
     andInputLocation:p18 andOutputQueueSize:999
     andOutputLocation: p20.
term:= Terminator newWithName:' Final Terminator'.

routing1:= Routing new.
routing1 addOperation: input key: nil
         processingTime: [:rg |rg normalMu: 30 sigma: 6]
         setupTime: nil;
         addOperation: machine2 key: nil
         processingTime: [:rg |rg normalMu: 100 sigma: 20]
         setupTime: nil;
         addOperation:machine5 key: nil
         processingTime: [:rg |rg normalMu: 60 sigma: 10]
         setupTime: nil;
         addOperation:machine7 key: nil
         processingTime: [:rg |rg normalMu: 110 sigma: 15]
         setupTime: nil;
         addOperation:machine8 key: nil
         processingTime: [:rg |rg normalMu: 110 sigma: 15]
         setupTime: nil;
         addOperation:machine6 key: nil
         processingTime: [:rg |rg normalMu: 120 sigma: 15]
         setupTime: nil;
```

```
        addOperation:machine9 key: nil
        processingTime: [:rg |rg normalMu: 150 sigma: 15]
        setupTime: nil;
        addOperation: term key: nil.
routing2:= Routing new.
routing2 addOperation: input key: nil
        processingTime: [:rg |rg normalMu: 60 sigma: 10]
        setupTime: nil;
        addOperation:machine3 key: nil
        processingTime: [:rg |rg normalMu: 90 sigma: 15]
        setupTime: nil;
        addOperation:machine4 key: nil
        processingTime: [:rg |rg normalMu: 100 sigma: 15]
        setupTime: nil;
        addOperation:machine2 key: nil
        processingTime: [:rg |rg  normalMu: 100 sigma: 15]
        setupTime: nil;
        addOperation:machine5 key: nil
        processingTime: [:rg |rg  normalMu: 70 sigma: 10]
        setupTime: nil;
        addOperation:machine6 key: nil
        processingTime: [:rg |rg normalMu: 100 sigma: 15]
        setupTime: nil;
        addOperation:machine9 key: nil
        processingTime: [:rg |rg normalMu: 90 sigma: 15]
        setupTime: nil;
        addOperation: term key: nil.

workOrder1:= WorkOrder newWorkOrderType: 'Work Order 1'.
workOrder2:= WorkOrder newWorkOrderType: 'Work Order 2'.
workOrder1 addComponentWFI:'part 1' andCWFIRouting:routing1.
workOrder2 addComponentWFI:'part 2' andCWFIRouting:routing2.
creater1:= WOCreator newWithWorkOrder: workOrder1
          timeBetweenCreationsGenerator:(NormalDist
                                          newMu:50 sigma:10).
creater2:= WOCreator newWithWorkOrder: workOrder2
          timeBetweenCreationsGenerator:(NormalDist
                                         newMu:200 sigma:20).
calendar schedule: [creater1 create] at: 0.
calendar schedule: [creater2 create] at: 0.
calendar schedule: [calendar clearStatistics] at: 6000.
calendar schedule: [calendar end] at: 10000.
calendar eventInitiator]
```

APPENDIX H: Smalltalk Simulation Output of Case Study 2

Simulation Output: Run 1 of 25

Calendar Statistics

Event List Length Information

Time of initialization = 6000.00

Current Time = 10000

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
12.1136	0.8851	9.0000	9.0000	15.0000	5075

<<< O >>>

Final Terminator Statistics

Time In System Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
22	6642.7244	923.5348	8160.4718	5109.5778	8160.4718

<<< O >>>

Machine Statistics

input (an Input Station Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
102	35.3056	13.6556	16.6707	10.6715	96.2289

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.9088	0.2878	0.0000	0.0000	1.0000	52

Number of Balks Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	0.0000	1

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.5344	0.5787	0.0000	0.0000	2.0000	154

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
102	21.1375	20.5245	21.7495	0.0000	72.7172

outputQueue information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
130.6298	18.8763	165.0000	97.0000	165.0000	139

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
35	5140.0646	729.3806	6333.2614	3949.0993	6333.2614

<<< O >>>

m2 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
35	105.1402	22.1047	92.6116	55.7802	157.4573

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.9261	0.2616	1.0000	0.0000	1.0000	23

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.2913	0.4634	0.0000	0.0000	2.0000	49

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
35	33.2891	34.2337	0.0000	0.0000	111.0621

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
1.1048	0.6445	2.0000	0.0000	3.0000	70

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
34	123.7201	59.0412	269.7157	4.8188	269.7157

<<< O >>>

m3 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
7	91.4410	10.4055	98.1300	74.0241	102.2006

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.1600	0.3666	0.0000	0.0000	1.0000	15

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	0.0000	1

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
7	0.0000	0.0000	0.0000	0.0000	0.0000

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.8158	0.3876	1.0000	0.0000	1.0000	15

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
7	469.1871	66.0409	401.9829	392.4992	608.8848

<<< O >>>

m4 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
7	91.4485	19.3716	68.1145	68.1145	122.3148

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.1600	0.3666	0.0000	0.0000	1.0000	15

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	0.0000	1

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
7	0.0000	0.0000	0.0000	0.0000	0.0000

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.1003	0.3004	0.0000	0.0000	1.0000	16

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
8	52.4019	29.2856	27.3994	19.7837	92.8305

<<< O >>>

m5 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
34	64.8660	11.5666	65.9354	40.5375	91.8436

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.5538	0.4971	1.0000	0.0000	1.0000	69

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	0.0000	1

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
34	0.0000	0.0000	0.0000	0.0000	0.0000

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
24.1671	3.5435	29.0000	17.0000	31.0000	58

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
23	2767.2892	461.0305	3528.0819	2046.1797	3528.0819

<<< O >>>

m6 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
22	117.1942	14.1369	104.7525	91.9440	162.7219

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.6540	0.4757	0.0000	0.0000	1.0000	38

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0275	0.1636	0.0000	0.0000	1.0000	9

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
22	5.0022	13.8251	0.0000	0.0000	62.0206

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
1.3185	0.5216	2.0000	0.0000	2.0000	46

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
22	240.3534	54.3614	174.7350	140.8197	300.8354

<<< O >>>

m7 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
18	111.3404	15.7066	102.8132	88.0459	140.9692

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.4771	0.4995	1.0000	0.0000	1.0000	36

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	0.0000	1

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
18	0.0000	0.0000	0.0000	0.0000	0.0000

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
1.5229	0.4995	1.0000	1.0000	2.0000	36

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
18	342.6720	54.6005	396.0353	273.5952	418.3360

<<< O >>>

m8 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
17	114.9948	15.9933	106.6089	80.1362	140.9536

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.4887	0.4999	0.0000	0.0000	1.0000	35

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	0.0000	1

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
17	0.0000	0.0000	0.0000	0.0000	0.0000

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.5113	0.4999	1.0000	0.0000	1.0000	35

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
17	111.2178	40.1779	79.4820	63.0072	194.1832

m9 (a Single Queue, Multiple Server Processing Object)

Processing Times Information

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
22	142.2016	23.8653	141.1253	88.5854	170.6517

Utilization Information

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes

| --- | --- | --- | --- | --- | --- |
| 0.7849 | 0.4109 | 1.0000 | 0.0000 | 1.0000 | 27 |

InputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0771	0.2668	0.0000	0.0000	1.0000	19

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
22	14.0218	22.0613	42.7196	0.0000	66.3448

OutputQueue Information

Queue Length Statistics

Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
0.0000	0.0000	0.0000	0.0000	1.0000	45

Time In Queue Statistics

Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
22	0.0000	0.0000	0.0000	0.0000	0.0000

<<< O >>>

AGV Utilization Statistics

AGV1

States	Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
Stop	0.2505	0.4333	0.0000	0.0000	1.0000	255
Charging	0.0000	0.0000	0.0000	0.0000	0.0000	1
MovingEmpty	0.2811	0.4495	0.0000	0.0000	1.0000	88
MovingLoaded	0.4684	0.4990	1.0000	0.0000	1.0000	168

BreakDown Time	Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
	none	none	none	none	none	none

AGV2

States	Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes

Stop	0.2385	0.4262	0.0000	0.0000	1.0000	205
Charging	0.0000	0.0000	0.0000	0.0000	0.0000	1
MovingEmpty	0.1172	0.3217	0.0000	0.0000	1.0000	46
MovingLoaded	0.6443	0.4787	1.0000	0.0000	1.0000	160

BreakDown Time	Total Obs.	Avg Obs.	Std Dev	Last Obs.	Min Obs.	Max Obs.
	none	none	none	none	none	none

<<< 0 >>>

Segment Utilization Statistics

Name	Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
s1	0.1658	0.3719	0.0000	0.0000	1.0000	85
s2	0.1624	0.3688	0.0000	0.0000	1.0000	85
s3	0.0833	0.2764	0.0000	0.0000	1.0000	85
s4	0.0881	0.2835	0.0000	0.0000	1.0000	85
s5	0.0991	0.2988	0.0000	0.0000	1.0000	86
s6	0.1508	0.3578	1.0000	0.0000	1.0000	86
s7	0.0000	0.0000	0.0000	0.0000	0.0000	1
s8	0.0000	0.0000	0.0000	0.0000	0.0000	1
s9	0.0000	0.0000	0.0000	0.0000	0.0000	1
s10	0.0000	0.0000	0.0000	0.0000	0.0000	1
s11	0.0000	0.0000	0.0000	0.0000	0.0000	1
s12	0.0803	0.2718	0.0000	0.0000	1.0000	81
s13	0.1547	0.3617	0.0000	0.0000	1.0000	81
s14	0.0755	0.2643	1.0000	0.0000	1.0000	80
s15	0.0000	0.0000	0.0000	0.0000	0.0000	1
s16	0.0000	0.0000	0.0000	0.0000	0.0000	1
s17	0.0366	0.1877	0.0000	0.0000	1.0000	79
s18	0.0731	0.2603	0.0000	0.0000	1.0000	79
s19	0.0712	0.2571	0.0000	0.0000	1.0000	45
s20	0.0520	0.2220	0.0000	0.0000	1.0000	45
s21	0.0656	0.2476	0.0000	0.0000	1.0000	35
s22	0.0340	0.1812	0.0000	0.0000	1.0000	35

s23	0.1184	0.3231	0.0000	0.0000	1.0000	80

<<< 0 >>>

Control Point Utilization Statistics

Name	Avg Value	Std Dev	Curr Value	Min Value	Max Value	No. Changes
p0	0.0000	0.0000	0.0000	0.0000	0.0000	1
p1	0.0525	0.2230	0.0000	0.0000	1.0000	85
p2	0.1035	0.3046	0.0000	0.0000	1.0000	153
p3	0.0210	0.1434	0.0000	0.0000	1.0000	97
p4	0.0510	0.2200	0.0000	0.0000	1.0000	85
p5	0.0000	0.0000	0.0000	0.0000	1.0000	85
p6	0.0225	0.1483	0.0000	0.0000	1.0000	87
p7	0.0000	0.0000	0.0000	0.0000	0.0000	1
p8	0.0000	0.0000	0.0000	0.0000	0.0000	1
p9	0.0000	0.0000	0.0000	0.0000	0.0000	1
p10	0.0345	0.1825	0.0000	0.0000	1.0000	81
p11	0.0660	0.2483	0.0000	0.0000	1.0000	125
p12	0.0540	0.2260	0.0000	0.0000	1.0000	117
p13	0.0000	0.0000	0.0000	0.0000	1.0000	79
p14	0.0000	0.0000	0.0000	0.0000	0.0000	1
p15	0.0000	0.0000	0.0000	0.0000	1.0000	79
p16	0.0000	0.0000	0.0000	0.0000	1.0000	79
p17	0.0510	0.2200	0.0000	0.0000	1.0000	69
p18	0.0330	0.1786	0.0000	0.0000	1.0000	45
p19	0.0000	0.0000	0.0000	0.0000	1.0000	79
p20	0.0000	0.0000	0.0000	0.0000	0.0000	1

<<< 0 >>>

Complete outputs are available from Dr. Terrence G. Beaumariage at Oregon State University

APPENDIX I: Smalltalk Classes and Code for Material Handling Extensions

Complete code is available from Dr. Terrence G. Beaumariage at Oregon State University